**Universidade de Aveiro** Dep. de Electrónica, Telecomunicações e Informática
**2007**

**Samuel de Sousa Silva**

# POLYMECO: Uma Ferramenta de Análise e Comparação de Malhas Poligonais

# POLYMECO: A Polygonal Mesh Analysis and Comparison Tool


**Universidade de Aveiro** Dep. de Electrónica, Telecomunicações e Informática
**2007**

**Samuel de Sousa Silva**

# POLYMECO: Uma Ferramenta de Análise e Comparação de Malhas Poligonais

# POLYMECO: A Polygonal Mesh Analysis and Comparison Tool

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. Joaquim João Estrela Ribeiro Silvestre Madeira, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Dra. Maria Beatriz Alves de Sousa Santos, Professora Associadada com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Aos meus pais

**o júri**

presidente

Prof. Dr. António Manuel Melo de Sousa Pereira
Professor catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Adriano Martins Lopes
Professor auxiliar do Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Prof. Dr. Joaquim João Estrela Ribeiro Silvestre Madeira
Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dra. Maria Beatriz Alves de Sousa Santos
Professora associada com agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

**agradecimentos**

Esta dissertação é fruto do contributo dado, a diferentes níveis, por um conjunto de pessoas a quem gostaria de deixar algumas palavras de agradecimento.

Em primeiro lugar, agradeço aos meus orientadores, o Prof. Dr. Joaquim Madeira e a Prof. Dra. Beatriz Sousa Santos, pela paciência, ajuda, e constante apoio ao longo de todo o trabalho; ao Prof. Dr. Carlos Ferreira, do Departamento de Economia, Gestão e Engenharia Industrial da Universidade de Aveiro, a preciosa colaboração na análise estatística dos dados apresentados no capítulo 5; e ao Prof. Dr. Frutuoso Silva, do Departamento de Informática da Universidade da Beira Interior, a sua contribuição como utilizador do PolyMeCo, que permitiu o teste da ferramenta proposta num cenário de aplicação real e de onde resultaram importantes correcções e novas ideias.

Não posso também deixar de manifestar o meu agradecimento ao Instituto de Engenharia Electrónica e Telemática de Aveiro pelas condições de trabalho que me foram proporcionadas e por ter suportado, na íntegra, os custos do meu curso de mestrado.

E o ambiente de trabalho não seria o mesmo se não fossem os amigos que ao longo do tempo tive oportunidade de fazer no IEETA, companheiros de jornada e desvario, em busca de objectivos similares.

Mas o percurso que me trouxe até aqui não foi somente percorrido no meio académico. É também importante não esquecer as pessoas que fora dele contribuiram de forma considerável.

Antes de mais, um profundo agradecimento aos meus pais, irmã e cunhado, pela presença e acompanhamento constantes; e aos meus amigos André Oliveira e Tiago Sousa, caçadores de satélites e companheiros na observação dos anéis de Saturno, pela amizade ao longo deste tempo.

E a todos os outros amigos, companheiros de viagem, de conversa e leitura, que de alguma maneira contribuíram para o trabalho descrito nesta dissertação, deixo também um sincero agradecimento.

**palavras-chave** Malhas poligonais, comparação, visualização e análise de dados

**resumo**

Os modelos definidos usando malhas poligonais são usados em diversas áreas de aplicação para representar diferentes objectos e estruturas. Dependendo da aplicação, pode ser necessário processar esses modelos, por exemplo, para diminuir a sua complexidade (simplificação). Este processamento introduz diferenças, em relação ao modelo original, cuja avaliação é um passo fundamental para permitir escolher a sequência de operações e os métodos de processamento que permitam a obtenção de melhores resultados.

Apesar de algumas ferramentas de análise e comparação das características de malhas poligonais serem descritas na literatura, pouca atenção tem sido prestada à forma como os dados provenientes dessa análise e comparação podem ser visualizados. Para além disso, devem ser disponibilizadas várias funcionalidades de forma a permitir uma utilização sistemática destas ferramentas, assim como uma adequada análise e exploração dos dados fornecidos.

O PolyMeCo — uma ferramenta de análise e comparação das características de malhas poligonais — foi projectado e desenvolvido tendo em conta os objectivos acima referidos. Através de um ambiente integrado onde diferentes opções de visualização estão disponíveis e podem ser usadas de forma coordenada, o PolyMeCo permite aos utilizadores uma melhor compreensão dos dados resultantes da aplicação dos números de mérito disponibilizados.

Esta nova ferramenta foi usada com sucesso em dois trabalhos de investigação: (1) para comparar as características das malhas resultantes de dois algoritmos de simplificação de malhas poligonais, e (2) para testar a aplicabilidade dos números de mérito que disponibiliza como estimadores da qualidade de modelos poligonais, tal como percebida pelos utilizadores.

**keywords** Polygonal meshes, comparison, visualization and data analysis

**abstract**

Polygonal meshes are used in several application areas to model different objects and structures. Depending on the application, such models sometimes have to be processed to, for instance, reduce their complexity (mesh simplification). Such processing introduces error, whose evaluation is of paramount importance when choosing the sequence of operations that is to be applied for a particular purpose.

Although some mesh analysis and comparison tools are described in the literature, little attention has been given to the way mesh features (analysis) and mesh comparison results can be visualized. Moreover, particular functionalities have to be made available by such tools, to enable systematic use and proper data analysis and exploration.

PolyMeCo — a tool for polygonal mesh analysis and comparison — was designed and developed taking the above objectives into account. It enhances the way users perform mesh analysis and comparison, by providing an integrated environment where various mesh quality measures and several visualization options are available and can be used in a coordinated way, thus leading to greater insight into the visualized data.

This new tool has been successfully applied in two research works: (1) to compare between mesh simplification algorithms, and (2) to study the applicability of the provided computational measures as estimators of user perceived quality as obtained through an observer study.

# Contents

# Chapter 1

# Introduction

Polygonal meshes are used in a wide range of applications (e.g., Cultural Heritage and CAD systems). Sometimes, the original mesh model (e.g., built up from scanned 3D data) must be processed in order to suit a particular purpose. This processing step may consist of smoothing to eliminate surface noise, simplification to reduce complexity, compression to reduce storage size and transmission time, or even watermarking to protect the model against modifications or copyright violations, to name just the most common operations.

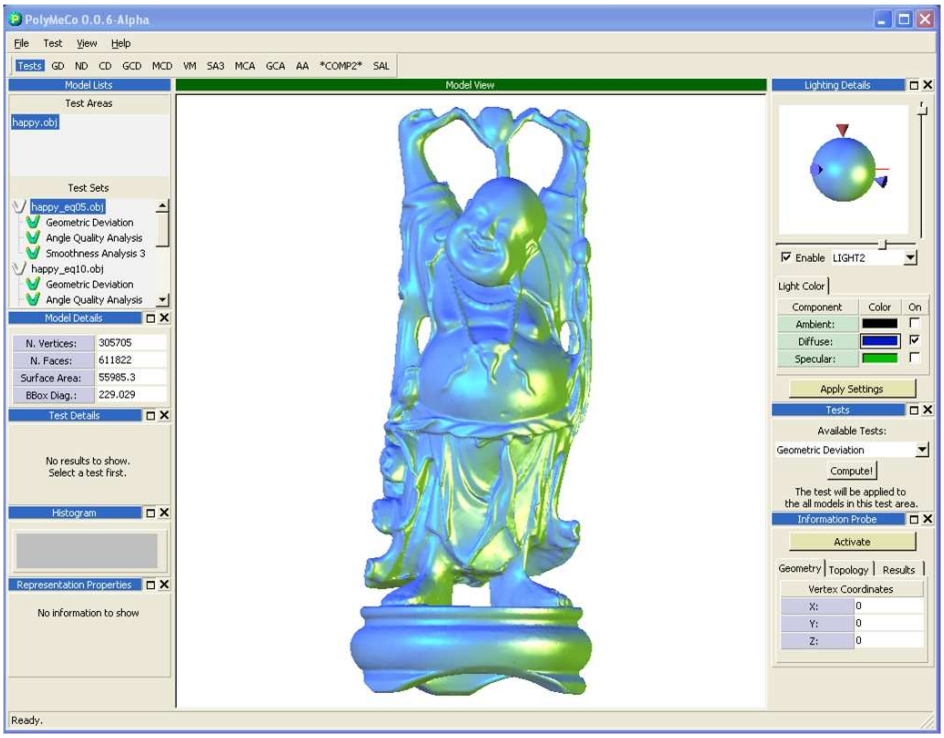



Figure 1.1: PolyMeCo's Graphical User Interface.

During the past years many methods have been proposed which allow mesh processing. Applying such methods to polygonal meshes introduces modifications which result in differences between the processed mesh and its original. These differences must then be evaluated in order to decide which method provides the best meshes and whether they are suitable for

the intended purpose.

There are some tools described in the literature, for mesh analysis and comparison, which have been used to compare the output of mesh processing methods in order to decide which one accomplishes the desired goal, while maintaining the smallest deviation regarding the original mesh (defined according to particular criteria, e.g., geometric distance). With a wide range of processing methods (for the same purpose, e.g., simplification) and a huge amount of possible applications, it is of paramount importance that validation be performed systematically to better understand the advantages and shortcomings of each method. This is particularly important, for example, in medical applications, where a confidence factor is a most desirable parameter in computer aided diagnosis systems.

## 1.1 Objectives

After analysing the mesh comparison tools described in the literature, and reviewing some of their usual applications, it was clear that they lacked several features deemed important. Those tools provide numerical descriptors, along with colored models which depict the data obtained using several computational measures.

But today, as the field of mesh processing has matured, it becomes necessary to have a tool which provides an enhanced environment for mesh analysis and comparison, where several models can be loaded during the same work session and data compared using proper visualization methods to allow, not only developers but also users in general, to test and choose among available mesh processing methods. It is important that such a tool provides a way of performing systematic analysis and comparison and allows exploring the obtained data leading to greater insight into the characteristics and effects of different mesh processing methods.

## 1.2 Developed Work

This dissertation describes a new tool for mesh analysis and comparison[1], called PolyMeCo (**Poly**gonal **Me**sh **Co**mparison), which provides an integrated environment (see Fig. 1.1) allowing not only the computation of several mesh features and difference measures (e.g., surface smoothness analysis and geometric distance) for mesh analysis and comparison but also the analysis of the obtained data performed using different visualization techniques (e.g., colored models and histograms). It allows working with several models in the same work session, which are organized in a hierarchical form, thus speeding up the process of mesh analysis and comparison for large sets of models. The user can switch easily between visualization modes, i.e., alternative ways of exploring the same data, ranging from simple mesh model rendering to extended views using colored models (several preset color scales are available), histograms and boxplots.

[1] It is not a mesh processing tool, i.e., it does not support modifying meshes in any way.

In order to store the obtained data for future reference or to continue the analysis at a later time, PolyMeCo allows saving the contents of the current workspace. This is particularly important: sometimes comparing two meshes can take a long time and, in this way, all the results are saved and easily viewed, at any time, with no additional delay.

PolyMeCo has already been used in some research activities, as described in Chapter 5, proving to be a step forward in its field.

## 1.3 Overview

Chapter 2 presents a short introduction to Data Visualization by describing a possible visualization pipeline and focusing on issues like the use of color to represent data or view transformations to enhance the visualization process.

In Chapter 3, an overview on Geometric Modeling using Polygonal Meshes is presented. It starts by describing polygonal meshes and their applications. Then, a possible mesh processing pipeline is described and an overview on some common mesh processing methods is presented, namely: smoothing, simplification, compression and watermarking. Next, some of the measures generally used to evaluate mesh features, as well as differences between meshes, are described. Finally, a brief analysis of several mesh comparison tools appearing in the literature is done.

Chapter 4 provides a description of the developed tool, PolyMeCo. It starts with a list of features deemed important for a mesh analysis and comparison tool. Then, it presents a possible mesh analysis and comparison pipeline (supported by PolyMeCo) along with all features available.

Application examples are presented in Chapter 5. It starts with a description of how PolyMeCo was used to compute several mesh comparison measures when aiming to find good estimators of user perceived quality, as obtained using an observer study to evaluate simplification methods. The second application example shows the results of the comparison between models simplified using two mesh simplification methods which was also performed using PolyMeCo.

Finally, Chapter 6 presents some conclusions and ideas for future work.

# Chapter 2

# Visualization

Visualization [29] is concerned with representing, manipulating and exploring data and information graphically in such a way as to gain understanding and insight into it, i.e., mapping of data to a visual form that supports human interaction in a workspace for visual sense making [33]. As such, there are several scientific domains which have important roles in Visualization, namely Computer Graphics, Human-Computer Interaction and Image Processing.

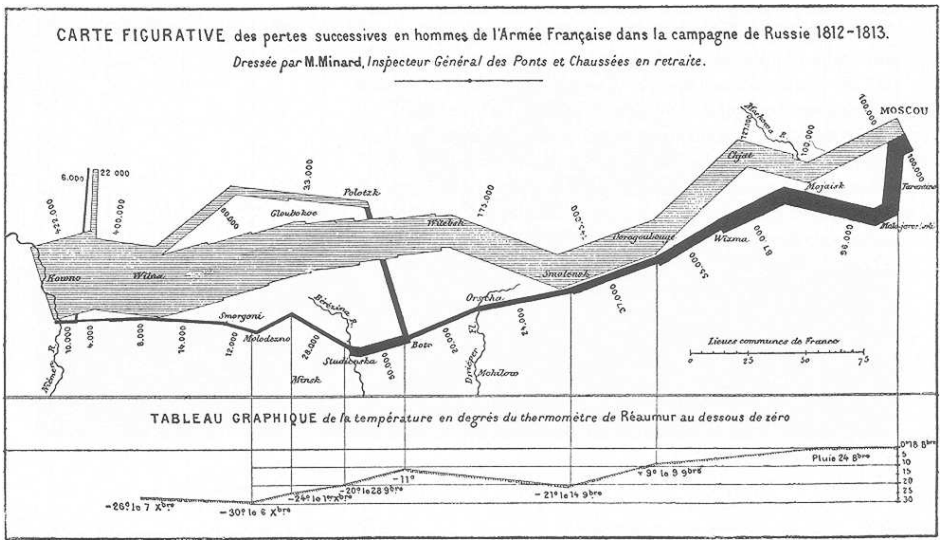



Figure 2.1: Napoleon's march to Russia by Charles Minard.

Across the centuries the advantages of visually representing large amounts of data were recognized. A well know example of early Visualization is that of Napoleon's march to Russia by Charles Minard (Fig. 2.1). Nowadays, with computers helping on the tasks of building and interacting with various data representations and different visualization techniques (available in software tools as the Iris Explorer [4] or libraries as the Visualization Toolkit [126]), the numerous application areas of Visualization range from Physics and Meteorology to Medicine.

Visualization is usually divided in two different areas: Data (or Scientific) Visualization and Information Visualization. The main difference between the two consists in the first dealing with inherently spatial data (e.g., temperature along an iron rod) while the second

deals with data not associated with any particular spatial distribution (e.g., results of a database query).

This chapter presents a brief overview on Data Visualization. After presenting the main stages of a possible Data Visualization pipeline, attention is focused on information deemed important for the work carried out, specifically the use of color and statistical tools to build data representations, and how these can be modified and augmented during the visualization process allowing greater insight into the data.

## 2.1 Data Visualization Pipeline

The Data Visualization process can be described by the pipeline presented in Fig. 2.2.

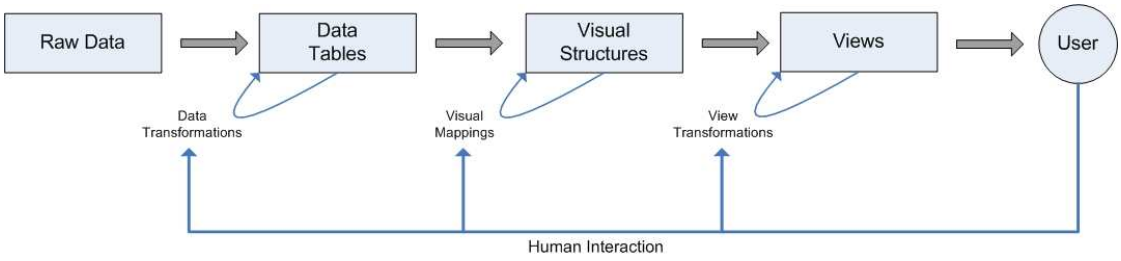



Figure 2.2: Data Visualization pipeline. Adapted from [33].

*Data Transformations*, i.e., transforming Raw Data into Data Tables imply, in general, loss or gain of information. At this stage issues like errors in the data or missing values are addressed in order to prepare the data to be visualized. Therefore, Data Tables often contain values or a structure derived from the Raw Data. Examples of derived data are statistical computations such as a mean value. Sorting variables or cases is an example of a derived structure. Data can be of different types, namely: nominal, ordinal, interval or ratio data.

*Visual Mappings* are the process through which data is mapped onto a Visual Structure. It is most desirable that this mapping allows a complete representation of the data without introducing unwanted data or artifacts. At this stage it is important to consider, among other aspects, the data type and goal of the visualization. Color mapping and statistical representations are typical visualization techniques used to build *Visual Mappings* and they will be addressed on the following sections.

*View Transformations* are used to interactively modify and augment representations. The most common *View Transformations*, namely Location Probes, Viewpoint Controls and Distortion, will be presented in Section 2.4.

## 2.2 Color Scales

The appropriate use of color in Data Visualization is a particularly important subject. The choice of the proper color scale to use with a particular data set is not just a matter of choosing the prettiest representation. Throughout the years researchers have studied this

subject and managed to propose guidelines which help users along the process of color scale selection. A brief overview on the subject of color scales is provided in what follows, focusing on the desired properties for color scales, the guidelines that should drive their choice, the advantages of applying those guidelines, the experimental research work on the field, and the tools proposed to help non-expert users.

### 2.2.1 Desired Properties for Color Scales

Having a sequence of numerical values $\{v_1 \leq v_2 \ ... \leq v_N\}$ represented by colors $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$, respectively, it is possible to identify the following desirable properties [85] [153] for a color scale:

- *Order*
  The colors chosen to represent the values must be perceived as having the same order as the values, i.e., if the values are ordered, the colors chosen to represent them must also seem ordered. An example can be the representation of a temperature scale by using the notions of *cold* and *warm* colors and their proportional mixtures in order to obtain a scale from cold to hot temperatures.

  It is important to note the special case of nominal data [118]: objects should be distinguishably different but, since they are not ordered, there should be no perceptual ordering in the representation.

- *Uniformity and Representative Distance*
  The color representation of two values should convey the distance between them and colors representing values which equally differ from each other should also seem equally different. Beyond that, it is required that clearly separated values must be represented by distinguishable colors and that close values must be represented by colors perceived to be closer. This is what Trumbo [153] calls the *Separation Principle*.
  When representing flow information, for example, complementary colors can be used to represent flows in opposite directions and similar colors (with slight differences) to represent flows in the same direction. Levkowitz et al. [85] identify analogous principles proposed by Pizer et al. [107] (*Associability*) and Robertson et al. [116] (*Separation*).

- *Boundaries*
  If there are no boundaries on the represented numerical data the color scale should not create this effect, i.e., the color scale must be able to represent continuous scales.

- *Rows and Columns Principle*
  This is one of the principles proposed by Trumbo [153] which applies only to bivariate information. It states that if it is important to preserve univariate information then the display parameters must not obscure one another, i.e., rows or columns having a constant value of one variable must have constant hue, saturation, or brightness. For example, using two display primaries (e.g., red and green) goes against this principle.

- *Diagonal Principle*

  The second principle proposed by Trumbo which only applies to bivariate information says that if the detection of positive associations of variables is a goal, the displayed colors must be easily identified as belonging to one of three classes: the ones near to the minor diagonal, the ones above it, and the ones below. This could be accomplished with the major diagonal made up of grays, elements of maximum saturation, or constant hue. A hue and brightness scheme violates the *Diagonal Principle* [118].

### 2.2.2 Univariate Color Scales

Univariate color scales map the value of a single scalar variable to a color representing that value. There are two kinds of color scales: the ones in which adjacent colors are similar to one another, forming a continuous path through a color space, and the ones that contain discontinuities, i.e., locations where adjacent colors are not similar at all. The color scales that will be briefly presented next (according to a survey presented on [113]), as they were considered as more suitable to the work being carried out, are continuous color scales.

1. **Color Model Components**

   - *Grey scale* – This color scale maps the value of a scalar to brightness (see Fig. 2.3a). In general, black represents the smallest value and white represents the largest, while shades of grey represent the values in between. This might happen differently with users that are more acquainted with printed data preferring a scale where an increasing value is represented by an increasing amount of ink thus mapping the lowest value to white and the highest to black. The efficiency of this color scale is enhanced due to the effectiveness of the human visual system at making judgments about shape from brightness variation which makes it suitable for tasks where understanding qualitative shape and pattern information is needed. This color scale exhibits a natural perceptual order (in brightness steps) and has a visual zero value (normally black).

     The problems in this color scale arise from the fact that it has a limited number of distinguishable display values and a limited contrast between different levels, making it less suited for tasks involving quantitative measures.

   - *Saturation scale* – In this color scale the value of a scalar is mapped to colors of increasing saturation, maintaining a constant hue (see Fig. 2.3b). Higher values are emphasized over lower values. Its main weakness is that it provides a limited number of distinguishable levels.

   - *Spectrum scale* – Also know as a rainbow scale (see Fig. 2.3c), it is formed by holding saturation and brightness constant while letting hue vary through its entire range. It follows the colors of the rainbow: first red, then orange, yellow, green, blue and violet. One of the problems that can arise with this scale is that some users see no intuitive ordering in the hues.

Normally, spectrum scales start at red and go through increasing wavelengths until violet. This can be an advantage for users having knowledge of the progression of visible light wavelengths. It may, however, present a problem due to the fact that the colors at the beginning and end of the scale (red and violet) are too close. So, it is usual to use this color scale limited to the red-blue range (see Fig. 2.3d) which allows a better differentiation of the extremes.

Another potential disadvantage of this color scale is that yellow (a very striking color) is in the middle of the scale. This can be a problem if one is interested in depicting extreme values because attention may be driven to the yellow areas.

Also, there are perceptual discontinuities along the scale which can lead, for example, to the perception of boundaries where none exist.

Although these problems and many other are described in the literature [119], the rainbow color scale is still the most used color scale in Visualization [25].

2. **Redundant Color Scales**

Using multiple display parameters to represent data may have the following advantages [113]:

- Each display parameter may convey a particular type of information better than any other (e.g., brightness to convey shape, hue for more accurately distinguishable display levels);
- Multiple display parameters may help overcome visual deficiencies. If one of the parameters becomes ambiguous due to a visual deficiency, another may compensate (e.g., use redundant hue and brightness);
- Multiple parameters reinforce each other making, for example, areas with differing values more visually different.

In Ware [157] the utility of redundant color scales has been experimentally confirmed leading to a suggestion that a color scale which varies in both luminance and hue can be used to accurately represent both metric and surface properties. Some examples are:

- *Redundant model components* – A straightforward redundant scale can be built by mapping data values to both hue and brightness (see Fig. 2.3e). This kind of scale has the advantage of being suitable for use by someone with dichromatic color deficiency.
- *Heated-object scale* – This scale represents a compromise between the grey scale and the spectrum scale. It goes from black to white passing through orange and yellow (see Fig. 2.3f). This color scale has a stronger perceived natural ordering than the rainbow scale, since it has a monotonic increase in brightness.
- *Optimal color scale* – This color scale was introduced by Levkowitz et al. [85] to describe a scale which maximizes the number of JNDs (just noticeable differences) while preserving a natural order.

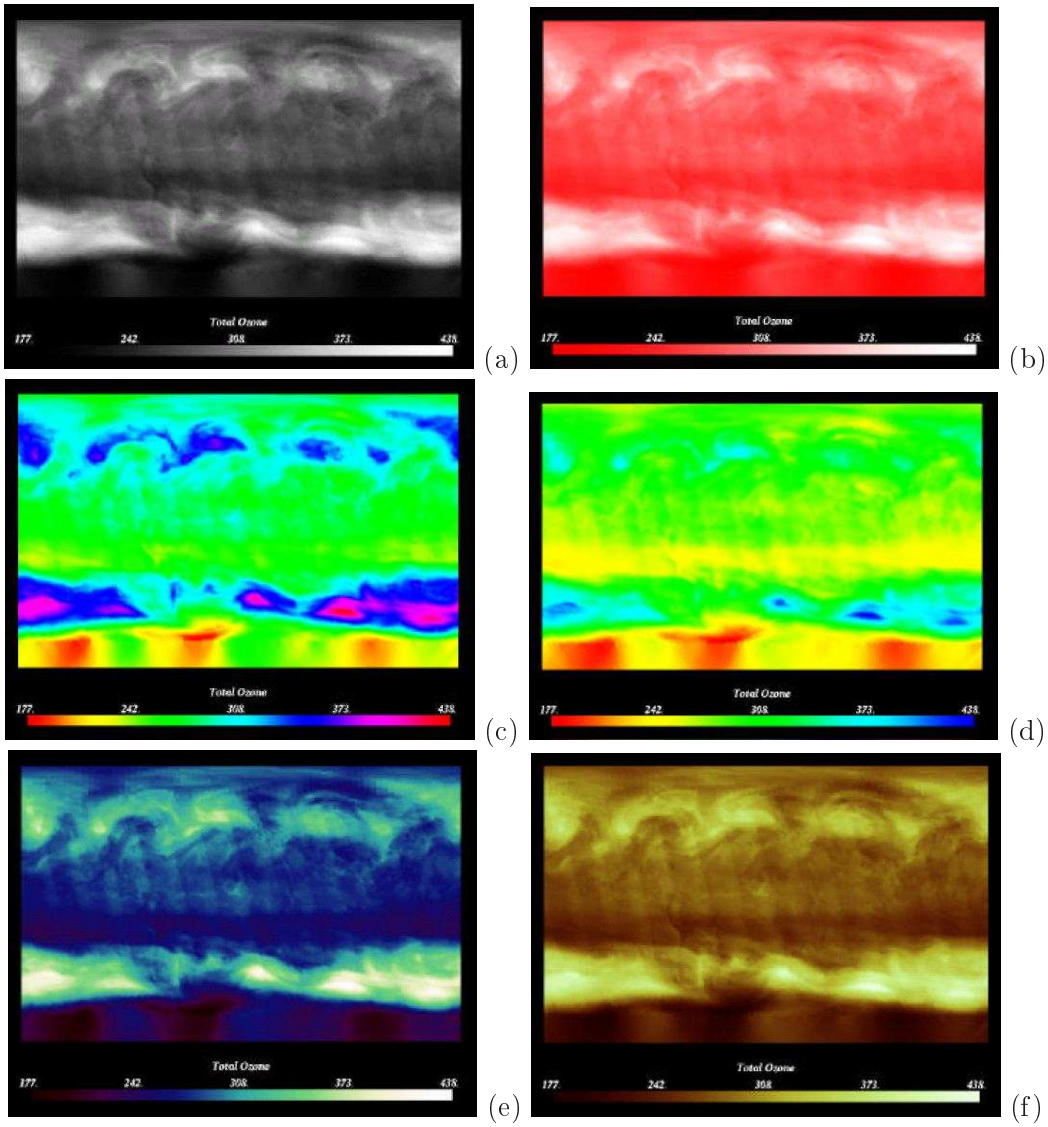



Figure 2.3: From left to right and top to bottom: grey scale, saturation, spectrum, limited spectrum, redundant hue-lightness and heated-object color scales applied to a data set [114].

3. **Double-Ended Color Scales**

   This kind of scale is created by joining two monotonically increasing scales at a common end point. For example, one can join a scale from grey to red and a scale from grey to blue building a scale from red to grey to blue. With such scales it is possible to visually represent high, low and middle values clearly, since they exhibit three distinct groups of colors.

### 2.2.3 Multivariate Color Scales

In a multivariate color scale two or more data variables are each mapped to a single color representing them. This is the same principle as the one used with redundant scales, but now each display parameter is related with a different variable.

Working with the RGB color model it is possible to map a variable into each one of its components, thus creating a multivariate color scale. For example, Landsat *false color* images are commonly produced by representing three multispectral scanner bands with levels of red, green and blue [115]. The result is that if the represented bands are highly correlated, then the image will be composed of shades of gray as the three components will have close values. This scheme has the advantage that the extremes of the variable range (black, red, green, blue) are easily detectable.

A problem occurs when one needs to decompose the shown colors into their components. How can similarities between areas that have the same value for two components but differ on the third, be detected?

An analogous scheme can be obtained using, for example, the HLS color model. For more details on the possible approaches see [113].

### 2.2.4 Color Scale Selection: Some Guidelines

Next, some important issues that have to be taken into account when selecting a color scale are presented.

1. **Data Type**

   When designing a visualization (picking a color scale) care must be taken in order that the most striking features of the image reflect the most important features of the data. If a representation catches the user's attention with unimportant data features, this may cause more interesting features to be missed [113]. Bright colors, sharp boundaries, or high saturation areas will most likely catch user's attention. Thus, it is important to consider the data that will be represented and its type, and know what is more important: for example, to call attention to middle values or to positive/negative deviations from a zero (threshold).

   Each type of data has its own particularities which should be addressed properly:

   - **Nominal Data**
     For nominal data no mathematical operations are possible, since the value assigned to a particular measurement represents a name or label. An example is the categorization of different lung diseases with numerical labels 1, 2, 3 and 4: no mathematical operation is meaningful on these data. As noted above, the used representation for this kind of data should not implicitly order it.
   - **Ordinal Data**
     With ordinal data, values are assigned to measurements (for example) but no assumption is made about the spacing in-between the measurements, i.e., there can exist a numbering of 1, 2, 3 and 4 but the distance between element 1 and 2 cannot be assumed to be equal to the distance between 3 and 4. Ordinal data are inherently discrete [120]. The used representation should allow discrimination between objects and the perception of their relative order.

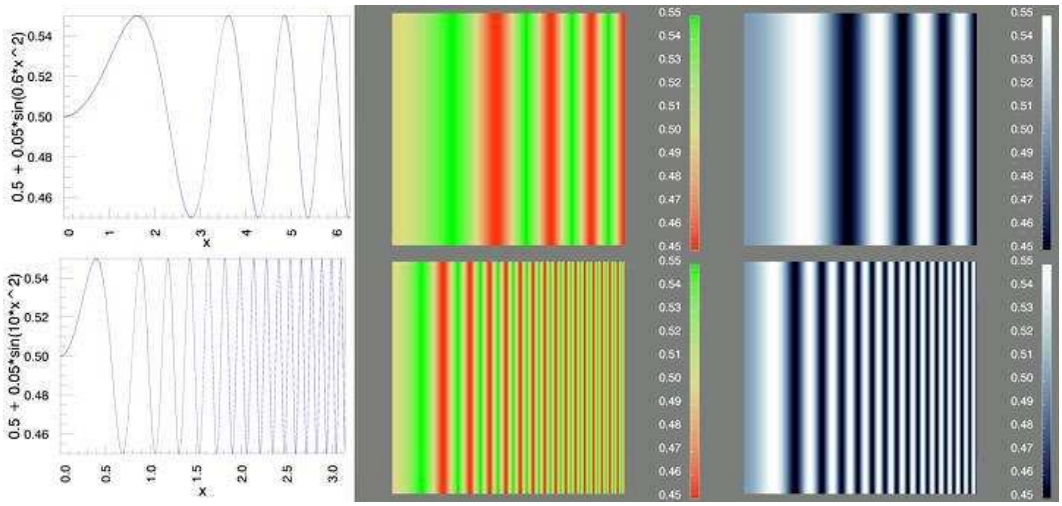



Figure 2.4: Two frequency modulated gratings represented using a saturation varying color scale (center) and a luminance varying color scale (right) [120].

- **Interval Data**
  In interval data, numerically equal distances between values are assumed to be actually equal. This kind of data is commonly a result of an experimental measure such as temperature, rainfall, etc. The used representation should account for this: equal steps in data values should correspond to equally perceived magnitude in the representation (resulting in what is called an isomorphic color scale).
- **Ratio Data**
  On ratio data, ratios between values are assumed to be equal and values increase/decrease monotonically about a true zero or threshold. This characteristic should be preserved in the representation. An example is the absolute temperature measured in Kelvin degrees.

2. **Spacial Frequency**

   An important issue to consider when choosing a color scale is human spatial vision. The luminance and saturation mechanisms in human vision represent an important role in spatial sensivity, but they have different characteristics. The human visual system accurately processes high-resolution images, or data which varies rapidly over an area if that spatial variation is represented as a variation in luminance, i.e., the luminance channels are responsible for processing high spatial frequency information. This means that when representing data with a high spatial frequency it is a good idea to use a color scale which provides a strong luminance variation across the data range. On the other hand, the saturation mechanisms in human vision are more sensitive to low spatial frequency variations. This kind of effects is illustrated in Fig. 2.4. On the top row a frequency modulated grating beginning at one cycle per image and increasing in spatial frequency is presented (the corresponding waveform is presented on the first column). The variation is represented using a saturation varying color scale (on the

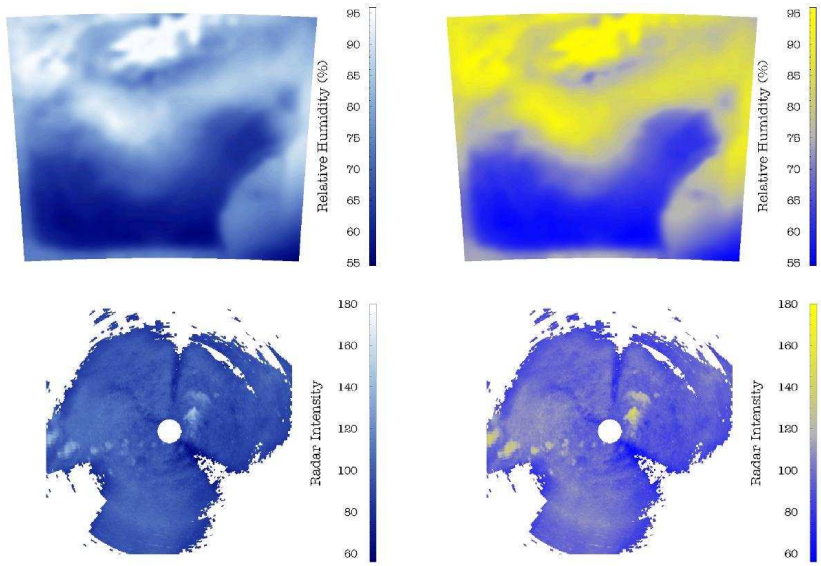



Figure 2.5: Isomorphic color scales for low (top row) and high (bottom row) spatial frequency data. The high frequency color scale (left) reveals more details in the bottom data, while the low frequency color scale (right) reveals more structure in the top data [118].

center) and a luminance-varying color scale (on the right). Notice how the saturation-variation color scale makes the sinusoidal variation more visible, at the low frequency end of the spectrum, than the luminance-varying color scale. On the bottom row the opposite thing happens: with the saturation-varying color scale (on the center) you can only observe the first few cycles of the frequency modulated grating, observing twice as many when using the luminance-varying color scale (on the right). Looking, for example, at interval and ratio-data, both luminance and saturation varying colormaps can produce the effect of having equal steps in data to be represented by equal perceived steps on the color scale, but the first will most certainly be more adequate to high spatial frequency data variations while the second will be more suited for low spatial frequency variations. Let us now look at how this reflects on real data. Figure 2.5 shows, on the top row, low spatial frequency data from a weather model (containing information about, for example, the variation in relative humidity over a geographic region) and, on the bottom row, high spatial frequency data from a radar scan (radial sweep from a weather radar sensor). The structure of the low spatial frequency data (on the top row) is practically lost when the data is represented with a map designed for high spatial frequency information (on the left). On the right, using a color scale for low spatial frequency results in a map which gives more information, specially in regions where humidity changes slowly over the geography. Notice how, in the lower right corner of the top right image, it is possible to clearly view the transition between high humidity and low humidity (yellow to blue) a feature almost undetectable when using the high spatial frequency data.

On the contrary, on the bottom row (high spatial frequency data), the usage of a high

spatial frequency color scale entails a good representation of the finely detailed structure of the data (even revealing sampling artifacts introduced by the sensor). On the other hand, the usage of the low spatial frequency color scale (on the right) gives poor results, not providing fine detail information and putting too much emphasis on the regions above the mean value (which appear in yellow).

3. **Task/Goals**

   The goal of a specific representation is paramount when choosing a color scale. Tasks which require the judgment of metric quantities in the data tend to work better with color scales which do not vary monotonically in the opponent color channels (brightness, red-green, yellow-blue). On the other hand, tasks involving qualitative judgments about value distribution shape are better served with color scales varying systematically in brightness, allowing our visual system to employ familiar shape-from-shading mechanisms [113].

   An early study by Tedford et al. [151] found a significant color-size effect leading to a conclusion that warm colors like red, orange and yellow appear larger than cool colors like green.

   In another study, by Cleveland et al. [39], users were asked to judge, on a map with equal colored areas in red and green, which one was the largest: the average observers considered the red areas where larger. The obtained results suggest also that the color-size effect grows stronger for very saturated colors, which indicates that these colors might not be the better choice for tasks where the user is expected to make judgments about size.

   Considering the kind of task to perform, the color scale can be designed accordingly. Next, examples for segmentation and highlight tasks are presented [118].

   - *Color Scales for Segmentation Tasks*

     Some of the rules used to create isomorphic color scales for ratio and interval data are also useful in creating maps for segmented data. In high spacial frequency data, luminance can be used to convey monotonicity; in low spacial frequency data monotonicity can be conveyed through the saturation component.

     In creating a segmented color scale it is necessary that the segments be each discriminably different from one another. This will limit the number of steps which can be represented. Bergman et al. [20] state that a higher number of steps can be effectively discriminated for low spatial frequency data than for high.

     An aspect to have in mind when dealing with ratio data (represented by a segmented map), where the zero value is semantically important, is that it is probably a good idea to have an even number of steps (with a transition at the zero level).

     On Fig. 2.6, on the left side, a five-level segmented color scale is used and, on the right side, a ten-level segmented color scale is used. They are applied to low

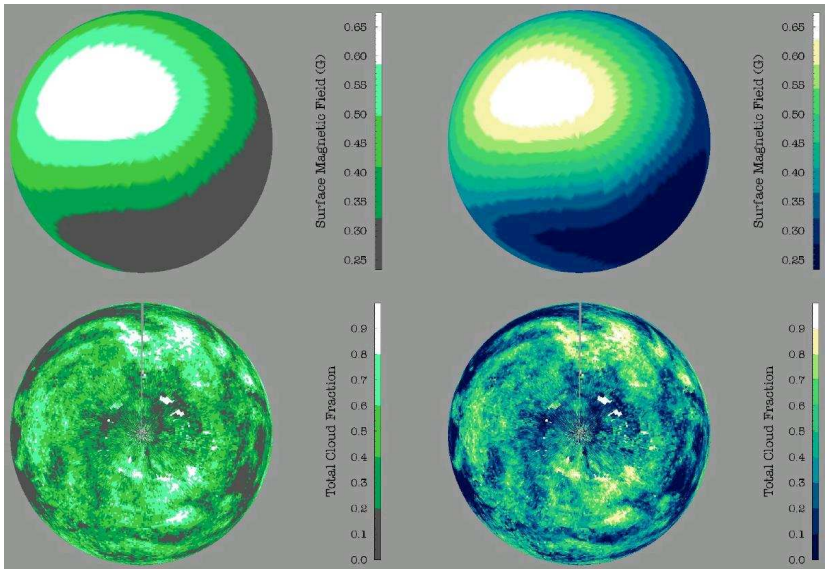



Figure 2.6: Segmented color scales applied to low (top row) and high (bottom row) spatial frequency data [118].

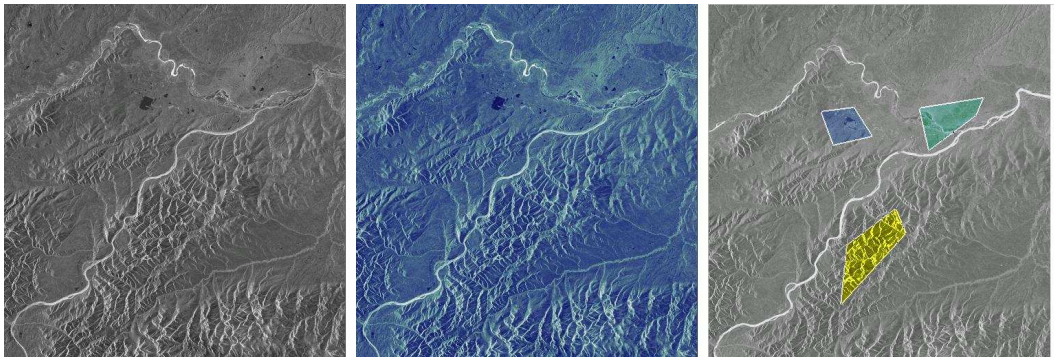


Figure 2.7: Two isomorphic color scales applied to a data set and a highlight scheme applied to the same data [118].

spatial frequency data (top) and high spatial frequency data (bottom). For the low spatial frequency data (top row), having additional levels provides additional information. For this particular case, additional features of the earth's magnetic field are revealed (notice the southern hemisphere). On the contrary, on the bottom row, showing high spatial frequency cloud fraction observations, additional features are not revealed by increasing the number of color scale steps.

- *Color Scales for Highlight Tasks*

  The principles which should guide the selection of a color scale for highlighting particular features in the data can be found, for example, in Julesz [73].

  Using these principles it is possible to design color scales which draw attention to a particular range in the data. Figure 2.7 shows an interesting approach using data from the visible part of the spectrum remotely-sensed from space. The two images on the left show the data using two isomorphic colormaps designed for high spatial

frequency data. The right image shows how color can be used to highlight a region of interest without interfering with other important aspects of the data. Across the image the luminance component of the colormap is identical, but, in the regions of interest, the hue component is modified producing three differentiable regions: one blue, one green and one yellow.

4. **Audience and Cultural Connotations**

   It is important to have information about the target audience of a visualization as it can give some clues for the color scale design. For example, conventions in one application area might place blue/violet colors of a spectrum scale at the low end (in order of increasing wavelength) while in another they may be placed at the high end (in order of increasing frequency) [113]. So, paying attention to area conventions may turn the process of designing the color scale easier and help avoid unintentional breaks with viewer expectations.

   Another issue is the way color tends to have strong cultural connotations varying from culture to culture. Following these conventions it is possible, for example, to reduce the cognitive load on the viewer or use connotations that suggest natural linkings between a variable or a variable value and the color used to represent it. For example, for an USA audience the color green is connected/associated with the color of money; a natural connotation, when visualizing temperatures can be that of high temperatures represented in red and low temperatures in blue.

5. **Visualization Type**

   It is important to consider the whole visualization during the process of color scale design for the individual elements. For example, three-dimensional visualizations have different constraints than those imposed by two-dimensional visualizations. A good example of the problems that can occur is related with shading: users use shading cues to judge the 3D shape of a representation object (e.g., an isosurface); a brightness varying color scale might interfere with the brightness values resulting from the shading calculations. Nonetheless a brightness varying scale may be used in planar objects on a 3D scene. Another issue can be posed by the requirement of displaying multiple variables in the same visualization. The used color scales should not generally overlap, with the representation for each variable interfering with the others as little as possible.

### 2.2.5 Learning Through Experimentation

In order to apply theoretical principles coming from other areas (such as psychophysics), verify the applicability of new principles, and find clues for the definition of new ones, many researchers have been conducting studies (namely, observer studies) [83].

Human color vision, a subject well studied (for more than a century now), provides strong clues for using color in visualization. However, the choice of colors for a particular task is more difficult, as it is far more complex, than the simple displays used by the experimental

psychologists. So, experiments are necessary to fill the gap between theory and practice [83]. The main goal is to use well established theories to build design guidelines and then use an experiment to validate the guidelines in an applied setting.

The work of Rogowitz et al. [117] presents a method which uses visual judgments to perceptually evaluate color scales. Since the literature points out that color scales which monotonically increase in luminance are good candidates for representing the magnitude of continuous data, the proposed method was designed to identify scales that include a monotonic luminance component.

The obtained results show that the proposed method might function as a quick procedure for identifying color scales with monotonically increasing luminance, with the advantage that it does not require display calibration or lengthy psychophysical procedures.

The work of Kindlmann et al. [80] is similar to the one presented by Bergman et al. [20]. They address the problem of color scale luminance control by proposing a novel technique for luminance matching. Their technique, given a fixed reference color, and a test color with brightness varied by the user, allows matching the luminance of both. They use images of faces in their experiment since they want to take advantage of the human characteristic of being good at recognizing faces, due to brain circuitry dedicated to this process [21]. The authors claim their method provides very good results and enables the creation of color scales, with any pre-determined pattern of luminance, in devices (e.g., monitors) which are not calibrated.

Other examples are the work of Ware [157], which leads to some rules guiding the process of color scale construction; of Healey [58], which presents a technique for effectively choosing multiple colors for use during data visualization; and of Montag [95], where the performance in judging values in univariate maps encoded using five different color scales is tested.

### 2.2.6 Auxiliary Tools and Methods

During the past few years some efforts have been made in order to provide users (in particular non-experts) with tools and methods which allow to select an appropriate color scale for their particular visualization purpose.

Rheingans et al. [114] propose a tool which allows the exploration of data sets by interactively manipulating the color scale. On the upper left of the screen appears the image space which shows the currently selected color scale applied to the data. In the center of the screen a 3D color space appears and a curve, within it, shows the path defining the sequence of colors composing the color scale. As the path in the 3D color space is modified, the image is dynamically changed accordingly.

Bergman et al. [20] present a tool called *PRAVDAColor* [1] which focuses on helping users to select color scales. With that purpose, they have built a library of color scales and defined a set of perceptual rules in order to select appropriate maps according to the structure of the data and visualization goal. They presented a taxonomy for color scale selection which guides their work. Table 2.1 presents that taxonomy: starting from the type and spatial frequency

[1] PRAVDA is the acronym for Perceptual Rule-Based Architecture for Visualizing Data Accurately.

of the data, and according to the representation task, several guidelines are proposed.

## 2.3 Statistical Representations

Statistical representations can also be used to provide a summary of a particular data set. The most commonly used are histograms and boxplots. A brief description of both is provided next.

### 2.3.1 Histograms

A histogram (see Fig. 2.8) is a graph showing a count of the data points falling in various ranges (i.e., bins). It can be used to provide a summary of a data distribution giving information about:

- The most common value.
- Type of distribution.
- Data symmetry and skewness.

The shape of a histogram can be particularly sensitive to the selected bin width. An example can be that of bimodal data (i.e., data with two values/ranges which appear often): if the bins are too wide this particularity may not be evident. On the other hand, if the bins are narrow, analysis becomes more difficult as the number of elements in each bin becomes smaller. So, to determine the correct bin size for a particular data set, several bin widths should be tested and its influence on the histogram shape observed. An example of a more elaborate method for optimal bin size selection can be found in Shimazaki et al. [130].

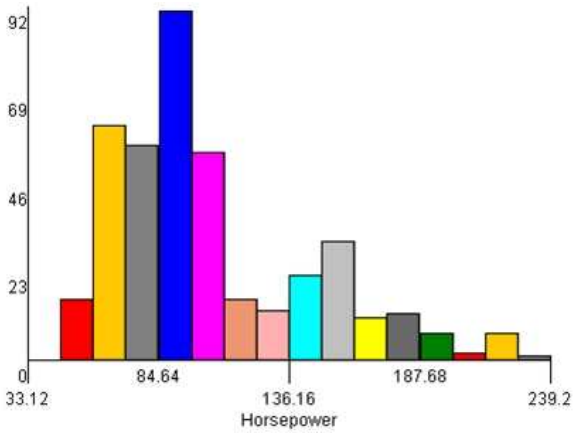



Figure 2.8: Histogram showing the horsepower of a set of 400 cars in the United States.

Another kind of histogram is the cumulative histogram. Instead of representing the number of elements in each bin, a sum of the number of elements present in all bins up to the current is provided. This means that the last bin will contain the total number of elements in the data set.

| Data Type | Spatial Frequency | Representation Task | | |
|---|---|---|---|---|
| | | **Isomorphic** | **Segmentation** | **Highlighting** |
| **Ratio (true zero)** | **Low** | *Luminance*: uniform<br>*Hue*: opponent complementary pair<br>*Saturation*: monotonically increasing from grey | Even number of segments<br>Many segments OK | Larger range for highlighted features |
| | **High** | *Luminance*: monotonically increasing<br>*Hue*: opponent or complementary pairs<br>*Saturation*: monotonically increasing from grey | Even number of segments<br>Fewer segments | Smaller range for highlighted features |
| **Interval** | **Low** | *Luminance*: uniform<br>*Hue*: opponent pairs<br>*Saturation*: monotonically increasing from grey | Many segments OK | Larger range for highlighted features |
| | **High** | *Luminance*: monotonically increasing<br>*Hue*: uniform or small hue variation<br>*Saturation*: monotonically decreasing | Fewer segments | Smaller range for highlighted features |
| **Ordinal** | **Low** | *Luminance*: uniform<br>*Hue*: variation around hue circle<br>*Saturation*: monotonically decreasing | Fewer segments | Increase luminance of highlighted areas |
| | **High** | *Luminance*: monotonically increasing<br>*Hue*: variation around hue circle *Saturation*: uniform | More segments | Increase saturation of highlighted area |
| **Nominal** | **Low** | *Luminance*: uniform<br>*Hue*: variation around hue circle<br>*Saturation*: uniform | Fewer segments than 7 | Increase luminance or saturation of highlighted areas |
| | **High** | | | |

Table 2.1: A taxonomy for color scale creation based on Data Type, Representation Task, and Principles of Perception [20].

The histogram is often used in combination with other statistical summaries such as the boxplot.

### 2.3.2 Boxplots

A boxplot [46] (see Fig. 2.9) is also a way of summarizing a set of data. It is a type of graph which is used to show the shape of the distribution, its central value, and variability.

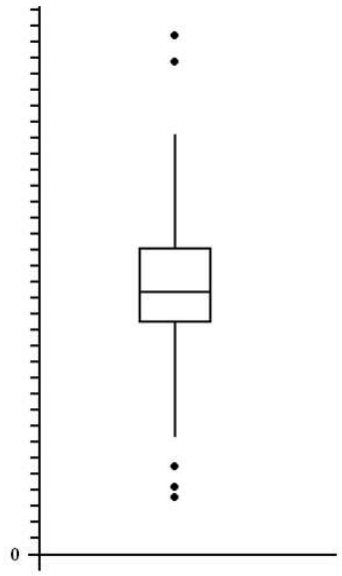



Figure 2.9: Boxplot example.

The box (see Fig. 2.9) contains the middle 50% of the data, i.e., values contained between the 25th percentile (lower edge of the box) and the 75th percentile (upper edge of the box). The line in the box represents the median value of the data. If this line is not equidistant from the upper and lower edged of the box then the data is skewed. The ends of the vertical lines (the "whiskers") represent the minimum and maximum data values, unless outliers exist in which case these lines extend to 1.5 times the height of the box (inter-quartile range). Finally, the points are outliers or suspected outliers.

Boxplots can be easily used to compare among data sets but they have the disadvantage of hiding distribution details. This can be overcome by using them with other tools like histograms.

## 2.4 View Transformations

*View transformations* are used to interactively modify and augment representations. There are three common view transformations [33]:

1. Location Probes

2. Viewpoint Controls

3. Distortion

### 2.4.1 Location Probes

*Location probes* are view transformations that use location in a representation to reveal additional information, i.e., users can obtain additional information about a particular element on the representation through, for example, a pop-up window in a way that is sometimes called details-on-demand [33]. Examples of location probes can be found throughout the visualization literature as in Gandhi et al. [47], where, in a tool to help users navigate the web by showing pages of a site in a tree view, probes are used to provide information about each node in the tree whenever the mouse is moved over it.

### 2.4.2 Viewpoint Controls

*Viewpoint controls* use affine transformations to zoom, pan and clip the view volume. These transformations are intended to enlarge the representation or change the observers position thus making the details more visible. When zooming into a particular representation it is important not to loose context, i.e., visualize any detail without loosing information about the general view. One solution [33] is to provide the user with a simple and fast zoom tool which allows switching from full view to detail and vice-versa rapidly. This, however, still requires the user to remember information not visible. Another solution can be to use "*overview + detail*". This technique consists in showing two windows: an overview of the representation and a detail window providing zoom of a particular area. The overview window provides context for the detail view, acting as a control window to change the region which is being zoomed.

### 2.4.3 Distortion

*Distortion* consists in a visual transformation which modifies a representation to create focus plus context views by combining overview and detail. It is more effective when the user can perceive the larger undistorted representation through the distortion. A well known example of distortion is the perspective wall [92].

Distortions are effective when the features or patterns, of use to the user, are not distorted in a way harmful to the task. For example, in the perspective wall the human perceives the linear sequence as folded, which means it is a distortion which leaves even the metric information invariant [33].

## 2.5 Conclusion

This chapter gave a short overview on Data Visualization, presenting a possible Visualization pipeline and focusing on issues deemed important for the work carried out, namely: the use of color and statistical tools to build data representations and the use of several *View Transformations* to enhance the visualization task.

The presented Data Visualization pipeline depicting the usual visualization task may be very helpful as a reference model when developing visualization tools.

Using color to transmit information is not a straightforward task. There are several issues which may influence the way information represented by color is perceived. The presented principles and rules may help selecting the proper color scale for a particular situation and detect problems that may arise.

*View Transformations* can help users to explore additional information. Given the fact that a visualization is presented in a limited space, it is important that methods are used which reveal information not visible in the current context. These allow the user to more easily deal with the information, accessing it in a hierarchical form.

# Chapter 3

# Geometric Modeling using Polygonal Meshes

Polygonal meshes are, nowadays, widely used in different application areas (see Fig. 3.1) of Computer Graphics and Geometric Modeling and Processing, as an alternative to polynomial spline surfaces, due to their simplicity, allowing, for example, easy model building by triangulating points obtained with 3D scanners.

Figure 3.1: Model of an angel represented using a polygonal mesh [9].

That wide range of applications has motivated the development of mesh processing methods, usually analogous to those used in signal processing [148], which allow mesh modification to meet specific criteria.

When processing meshes for a particular purpose, it is possible to define which method

to use and how, just like we would do, for example, when filtering an audio signal. This inter-method and inner-method variability requires that the results are assessed in order to choose the most adequate processing.

Along with the mesh processing methods, several operators have been developed which allow the computation of surface properties and mesh feature evaluation, e.g., curvature. Mesh comparison can then be performed by comparing correspondent mesh features between meshes. For this purpose, and in order to compare the quality of different mesh processing methods, some mesh comparison tools have been proposed.

This chapter provides a short overview on Geometric Modeling using polygonal meshes by presenting a possible mesh processing pipeline and information regarding some usual mesh processing operations. An overview of methods for mesh feature evaluation and mesh comparison ends the chapter.

More information on this subject is available in the Eurographis Tutorials presented by Kobbelt et al. [82] and Botsch et al. [27].

## 3.1 Polygonal Meshes

A 3D mesh $\mathscr{M}$ is defined [128] as a tuple $\{V, E, F\}$ of vertices $V = \{v_i | v_i \in \Re^3, 1 \leq i \leq m\}$, edges $E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$, and faces $F$ which are usually, for the sake of simplicity and performance, triangles $F = \{(v_i, v_j, v_k) | v_i, v_j, v_k \in V, i \neq j \neq k\}$, but may include other kinds of planar polygons. Figure 3.2 shows these different entities.

Meshes represented in this way are also known as boundary meshes in order to distinguish them from 3D volumetric meshes (e.g., tetrahedral) and emphasize the fact that they represent a 2D surface embedded in 3D.

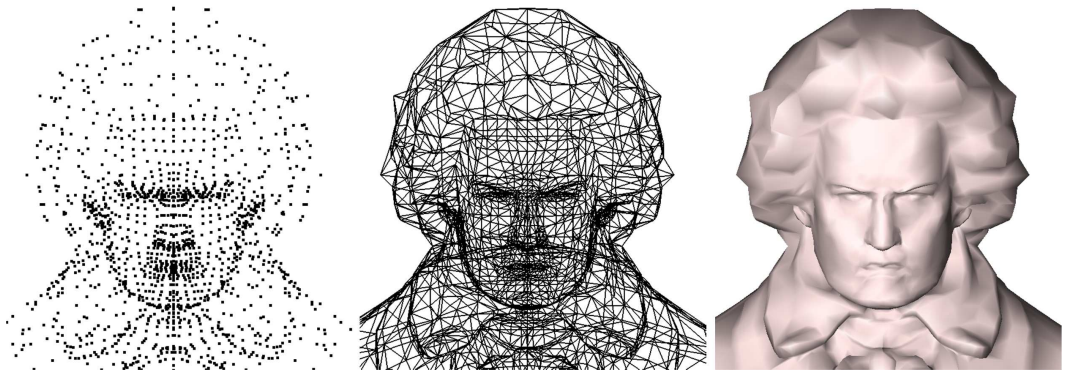


Figure 3.2: From left to right: vertices, edges and faces of a polygonal mesh.

The vertex description includes information regarding geometry ($x, y, z$ coordinates for each vertex) and, optionally, photometry (vertex normals, vertex colors or texture coordinates). Faces can also have photometry information associated, such as surface normals.

A usual mesh representation for storage may be that of a list of vertices and a list of faces defined by the indices of the vertices which compose them. For example, to represent the

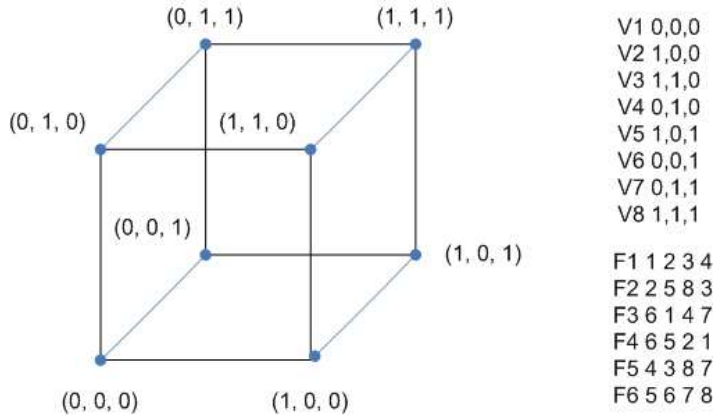



Figure 3.3: Defining a cube using a polygonal mesh. Left, a cube and the coordinates of each vertex. Right, list of vertices and list of faces (each face defined by 4 vertex indices) which define it.

cube on the left of Fig. 3.3 using a polygonal mesh, the two lists presented on the right would be enough.

This kind of representation (with a list of vertices and a list of faces) is typical in several file formats for mesh data storage like OBJ [5] and PLY [7]. Several constraints are to be considered in the relationship between mesh entities (vertices, edges, faces) in order to obtain valid representations. In general it is desirable that a mesh be two-manifold, i.e., each point on its surface is homeomorphic to a disk (half-disk at boundary points) [27]. A triangle

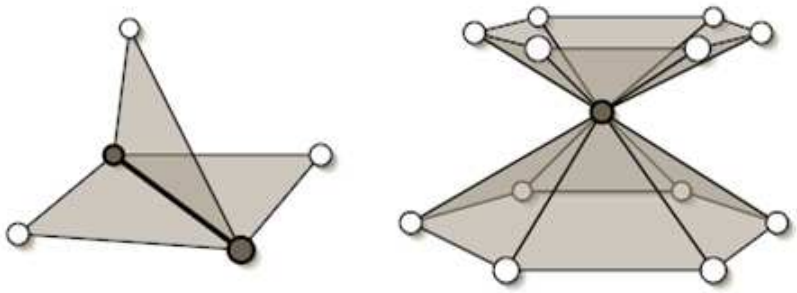


Figure 3.4: Examples of a non-manifold edge and a non-manifold vertex.

mesh is considered two-manifold if it does not contain non-manifold edges/vertices and has no self-intersections. A non-manifold edge has more than two incident triangles and a non-manifold vertex is, for example, the joining point of two separate surfaces in such a way that the vertex becomes incident to two triangle fans (see Fig. 3.4 for examples). Most mesh processing methods do not deal well with non-manifold meshes, since around non-manifold configurations there are no clearly defined neighborhoods.

### 3.1.1 Data Structures

In order to create and manipulate polygonal meshes it is necessary to have a proper data structure which allows fast and efficient access to the different entities (faces, edges and

vertices) and to local neighborhood information, e.g., the 1-ring (direct neighbors) of a vertex. As constant-size structures can be stored more efficiently, the restriction to triangle meshes or the use of edges as the topological primitive is usual [22]. An overview and comparison of several mesh data structures can be found in [76], and in [45] a short overview on data structures for non-manifold meshes is presented.

When choosing a mesh data structure it is important to analyse the topological and algorithmic requirements [27], i.e., which kind of meshes are to be represented (e.g., triangle or arbitrary polygonal meshes) and which operations need to be applied to them (e.g., just rendering, geometry modification, multi-resolution).

There are mainly two kinds of data structures: those which store connectivity information at face level, *Face-based*, and those which store connectivity information at edge level, *Edge-based*.

### Face-based

Each face (the basic topological unit) contains information about its vertices and adjacent faces. For each adjacent face, the index of the adjacent edge is also stored (see Fig. 3.5).

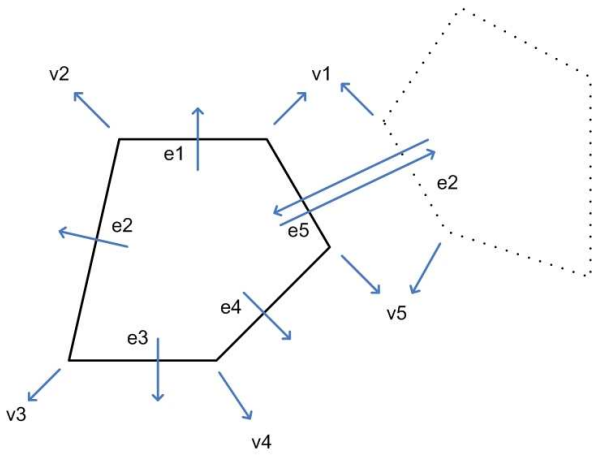



Figure 3.5: Face-based data structure. A face is described by an array of vertex pointers and an array of neighbor pointers (one such neighbor is indicated in dotted outline). Note that the neighbor has its own edge number assignment which may differ across the shared edge [167].

This kind of data structure is very convenient when working with subdivision and multiresolution hierarchies [167], since subdivision operations are performed using face and vertex information and there is no primary interest in edges.

### Edge-based

The most commonly used data structures to represent orientable two-manifold polygonal meshes [22] are the winged-edge [18] and the halfedge [160] structures.

The winged-edge data structure associates eight references to each edge: two vertices, two faces and four incident edges. This representation has the shortcoming of not providing

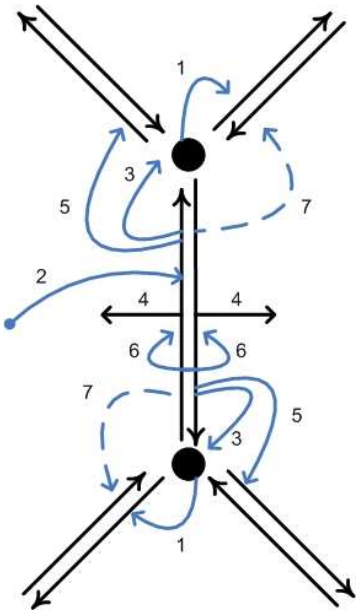



Figure 3.6: Halfedge data structure depicting how the information is stored. (1) halfedge starting at this vertex; (2) halfedge belonging to this face; (3) halfedge points to this vertex; (4) halfedge belongs to this face; (5) next halfedge in the face (counter-clockwise); (6) opposite halfedge; (7) previous halfedge in the face (optional).

information about edge orientation. The halfedge data structure solves this by splitting each edge in two neighboring halfedges. In a halfedge data structure each vertex references one outgoing halfedge, each face references one of the halfedges that bounds it and each halfedge has information associated with the vertex it points to, its next and opposite halfedges and the face it belongs to (see Fig. 3.6).

#### Other data structures

There are other, less used mesh data structures [22]: *quad edge*, which allows the representation of non-orientable manifolds; *radial edge*, to deal with non-manifold meshes; and *directed edges* [31], which is very memory efficient but only supports triangles. Recently, the *Adjacency and Incidence Framework (AIF)* [132] was presented. It is a data structure providing fast access and retrieval of local adjacency and incidence information. This is accomplished using a single indexed query operator, called mask operator, which allows access to connectivity information independently of mesh size. Unlike other boundary representation data structures, *AIF* is not oriented (but is orientable), i.e., it does not contain oriented cells (e.g., half-edges). Consequently, it is more concise.

### 3.1.2 Software Libraries

From the above mentioned data structures, halfedge is the most widely used and there are several libraries which provide its implementation and additional mesh processing operations. Some examples are:

- *Harvard Graphics Archive Mesh Library* [52] – This library, developed in C, provides an implementation of a halfedge data structure.
- *Computational Geometry Algorithms Library (CGAL)* [23] – This library, developed in C++, provides the implementation of several Computational Geometry algorithms (e.g., point-cloud triangulation). It also provides an halfedge data structure.
- *OpenMesh* [28, 6] – This library, developed in C++, has the main purpose of providing an halfedge structure and a set of features (e.g., vertex iterators) which allow the manipulation of a stored mesh. It is possible to work with a kernel for generic polygonal meshes or one specifically developed for triangular meshes. OpenMesh also provides a simplification module, which can be used to simplify meshes stored in its data structure, and allows the creation of multi-resolution meshes.

## 3.2 Application Areas

Polygonal meshes have, nowadays, a wide range of applications in several domains. In Cultural Heritage applications, mesh models have been used to represent huge statues [86, 53] or monuments, thus allowing the preservation of their features, measuring [125] (e.g., for restoration purposes) and close examination using virtual reality environments. Meshes have also been used to model archaeological sites and ancient building reconstructions [54], based on excavation data and the archeologist's interpretation. Figure 3.7 shows an example of polygonal mesh usage in Cultural Heritage.

Figure 3.7: Mesh application in Cultural Heritage. On the left a photograph of Michelangelo's David. On the right a computer rendering made from a geometric model created for the Digital Michelangelo Project [86].

In medical applications mesh models have been used in tools supporting diagnosis and treatment by providing the physician with, for example, 3D models of organs [68] or the skull [35] (e.g., for cranioplasty), allow virtual endoscopy [17] and support virtual surgery systems for training [142]. In the automotive industry 3D models of car bodies are used for testing purposes in order to inspect, for example, their surface curvature [146]. In computer games and animation films (Fig. 3.8), polygonal meshes are used to model characters and

Figure 3.8: *Cars*, the latest Pixar animation movie.

environments. In science and engineering, polygonal meshes are used to support simulations [89] (finite elements) and create visualization tools which allow additional insight into scientific data/phenomena [78]. Polygonal meshes are also used in CAD/CAM systems, with applications in several domains such as Mechanical Engineering and Architecture.

## 3.3 Polygonal Mesh Processing

Mesh processing is a key topic in Computer Graphics, Geometric Modeling and Computer Aided Design [140]. Nowadays, scanning techniques allow acquiring surfaces at a very high level of detail. These very complex (i.e., with a large number of faces and vertices) meshes require geometry processing such as filtering, to remove surface noise, simplification to reduce their complexity, compression, for efficient storage and streaming, or even watermarking to protect them against modifications or unauthorized copying. In what follows a brief overview on the mesh processing pipeline and on the above mentioned mesh processing tasks is done.

More detailed information on the subject can be found in the state-of-the-art reports by Taubin et al. [149] and Sorkine et al. [140].

### 3.3.1 Mesh Processing Pipeline

Mesh processing is usually described by the pipeline of Fig. 3.9 [27]. Raw data can be obtained by mechanical or optical scanning of an object, from volume data sets or from numerical simulations. Then, this point cloud is converted into a polygonal mesh using, for instance, triangulation methods and artifacts like holes or topological problems are corrected. At this stage the quality of the mesh can be evaluated in order to detect surface noise or badly formed triangles. The mesh can then be submitted to surface smoothing for noise removal, simplification for complexity reduction or remeshing for mesh quality improvement. All these operations are submitted to quality control, in order to understand the impact they have on the mesh and if the resulting meshes are acceptable for a particular application.

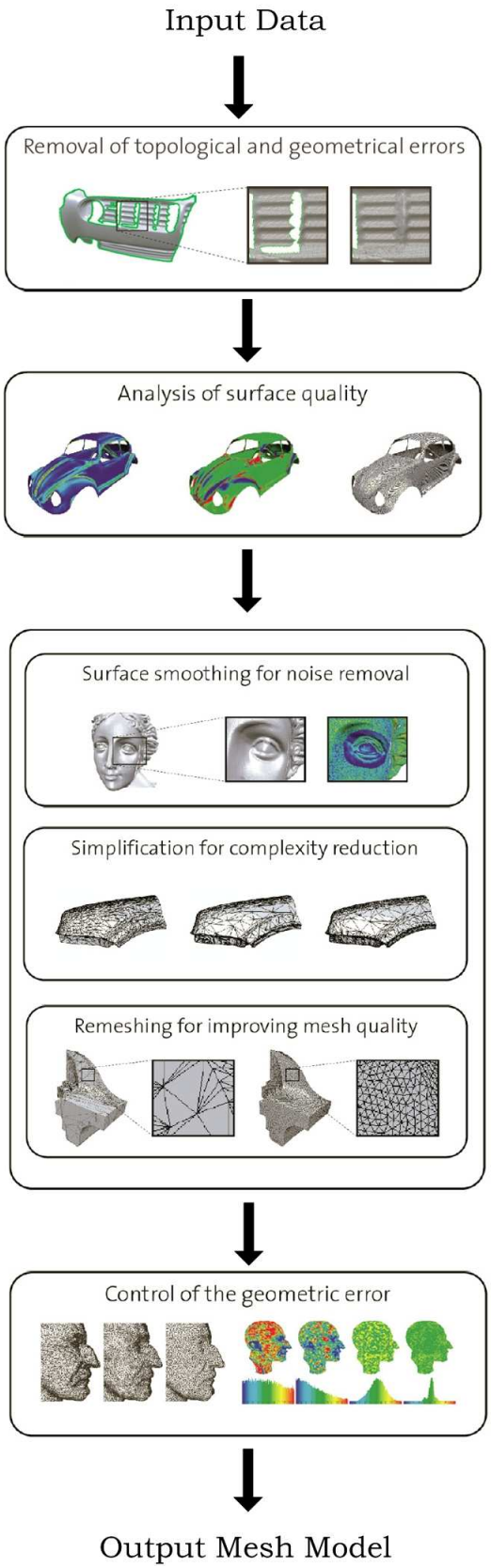



Figure 3.9: Mesh processing pipeline. Adapted from [27].

Figure 3.10: From left to right: original model, noisy model and smoothed model [129].

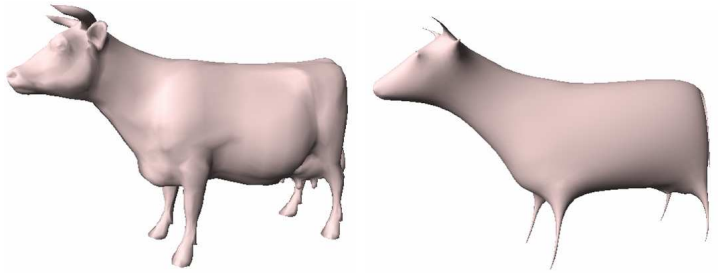


Figure 3.11: Mesh shrinkage: by applying several Laplacian smoothing iterations to a mesh, without any constraint, it tends to shrink, collapsing to its center of gravity.

### 3.3.2 Smoothing

Along the different stages of the mesh building/processing pipeline surface noise is introduced. This can happen, for example, during the data acquisition process, in intermediate calculations (due to limited precision) or in the data manipulation and surface reconstruction algorithms [129]. Removing or attenuating such noise can be accomplished by slightly modifying vertex positions in order to obtain a surface which is globally smoother. Figure 3.10 shows the results obtained after applying smoothing to a model.

In the past few years many surface smoothing algorithms have been proposed. Taubin [148] proposed a solution based on signal processing methods. His method was then improved by several authors, such as Kobbelt [81] and Desbrun et al. [43]. Even though such improved methods usually provide good results they can, sometimes, destroy important mesh features (e.g., edges and more detailed areas) [129]. A well-known example is that of mesh shrinkage when using Laplacian smoothing without constraints [149] (see Fig. 3.11 for an example). To preserve mesh features several methods have been proposed such as those of Taubin [150], Ohtake et al. [100], Jones et al. [71], Fleishman et al. [44] and, more recently, Shen et al. [129] (which includes a brief description of the previously referred methods) and Chen et al. [34].

### 3.3.3 Simplification

Mesh simplification algorithms decrease mesh complexity by reducing the number of their vertices (see Fig. 3.12, for an example), while trying to preserve as much of their shape and appearance as possible. In the past few years many simplification methods have been

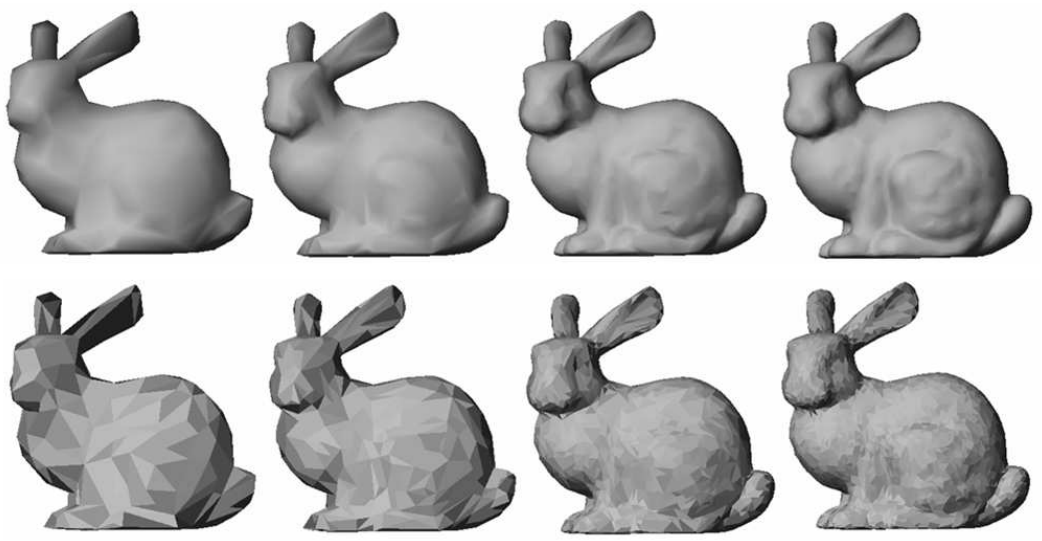


Figure 3.12: *Stanford Bunny* at different simplification levels: 250, 500, 1500 and 2500 vertices. Top, Gouraud shading; bottom, flat shading.

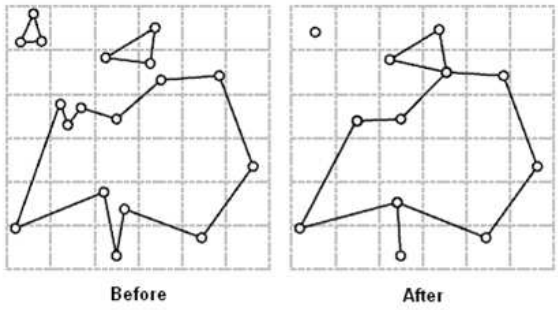



Figure 3.13: Vertex clustering example in 2D: the plane is divided into equally sized cells and all points belonging to the same cell are joined [59].

presented in the literature and a survey can be found in [90].

According to their characteristics, mesh simplification methods can be divided in several categories [72]: vertex decimation [127]; vertex clustering [122] (see Fig. 3.13 for a 2D example); edge contraction [62]; vertex pair contraction [49]; re-tiling [154]; and wavelet based simplification [51].

These vertex merge, removal and clustering operations can usually be encoded in a tree which can then be traversed in any direction, thus allowing a multi-resolution (level of detail) model. A few simplification methods have been proposed which use these multi-resolution models and simplification criteria based on the camera position on the scene (view-dependent simplification), or on the perceptual characteristics of the human visual system, providing a simplified version in run-time which is adequate to the visualization conditions. Examples are those of Hoppe [63], Kim et al. [79], Azuma et al. [15] and Luebke et al. [91].

Figure 3.14 shows a head model rendered at full complexity (top) and using view-dependent simplification (bottom). Both models look similar but the one on the bottom

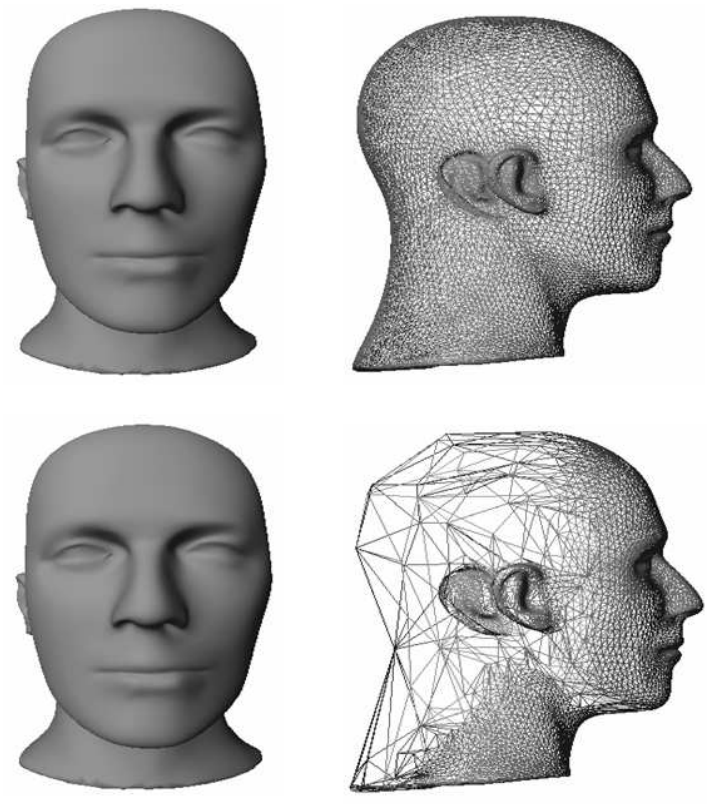


Figure 3.14: View-dependent simplification. Top, the original model; bottom, a model rendered using view-dependent simplification.

is rendered faster due to a smaller number of faces. By comparing the models in wireframe, it is clear that regions not viewed by the user where simplified to reduce complexity.

There are also methods which extend previous approaches by allowing the user to specify important features of the model [77, 108]. This information is then considered when selecting candidates (e.g., triangles) for simplification.

Recently, Jong et al. [72] and Jang et al. [69] proposed methods which provide enhanced ways of automatically identifying and dealing with model features by preserving them as much as possible or removing them consistently. An example of the latter can be seen in Fig. 3.15.

With mesh complexity growing at a fast pace, models do not fit into the main memory, which leads to intensive virtual memory usage. To avoid performance degradation, due to memory swapping, some out-of-core simplification methods have also been proposed. A few notable examples are those of Lindstrom [87], with improvements by Lindstrom et al. [88], and Wu et al. [161].

### 3.3.4 Compression

When dealing with polygonal mesh compression there are two kinds of information that can be compressed: geometry and connectivity [121].

In geometry compression, vertex coordinates are usually quantized (aiming to maintain a

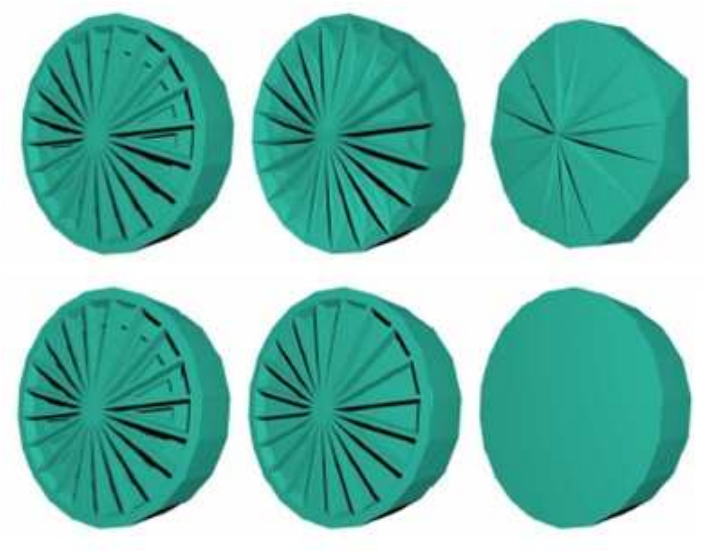


Figure 3.15: On top, three levels-of-detail obtained with QSlim and, on bottom, equivalent simplifications obtained with the simplification method proposed by Jang et al [69]. Some of the features could not be preserved, so they were removed in order to obtain a model offering better perceptual quality.

desired precision) and then efficiently coded using statistical codes. This operation is usually lossy, i.e., after compression it will not be possible to restore the original geometry, for example due to truncation during the quantization process. Beyond the typical approach of geometric coordinates quantization, there are methods which transform the geometric coordinates into a more suitable transform domain. The method proposed by Karni et al. [75] obtains the mesh spectral coefficients and then applies quantization to them. Another method, proposed by Sorkine et al. [141], transforms the spatial coordinates into what they call $\delta$-coordinates in order to shift the quantization error from high to low frequencies, thus turning it less noticeable.

In connectivity compression, connectivity information is efficiently coded. Generally, the method used has to be lossless, i.e., it must be possible to recover all the information during the decompression process. This is due to the fact that connectivity information is more important than geometry information mainly because it sets how faces are formed and provides neighborhood information, which is often used in geometry compression to obtain higher compression rates: instead of coding the coordinates for a particular vertex it is usual to code only their difference regarding a close neighbor (prediction methods). Beyond that, losing connectivity information would be unacceptable, for example, in application domains as CAD modeling.

Mesh compression algorithms can be divided in two groups: single-rate and progressive compression. In single-rate compression, geometry and connectivity information are compressed and decompressed as a whole. In progressive mesh compression a mesh is decomposed in a base (coarser) mesh and a sequence of refinements. This information is then encoded into a stream. The main advantage of progressive compression is that during decompres-

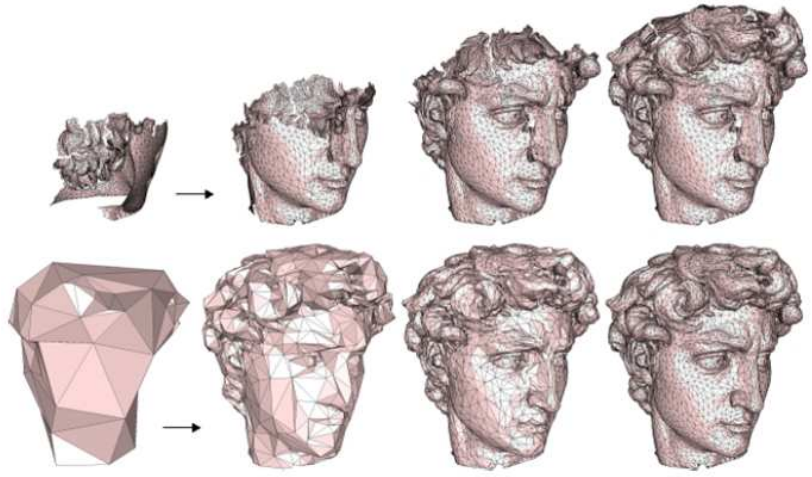


Figure 3.16: Single rate (top) vs. progressive compression (bottom) [12].

sion both connectivity and geometry are incrementally reconstructed thus allowing viewing a (coarse) complete version of the mesh right from the start (see Fig. 3.16 for a comparison between single-rate and progressive compression). This is also suitable to transmit information regarding huge models through a network.

A common operation in lossy mesh compression methods (both single-rate and progressive) is that of remeshing the model prior to compression. The original mesh is considered as just one of the possible representations of the shape geometry. This operation is performed in order to obtain a more regular mesh which will enable greater compression gains.

Some recent surveys about compression methods are presented by Alliez et al. [12] and Peng et al. [105].

### 3.3.5 Watermarking

One of the possible approaches to protect 3D models against illegal copying, tampering and copyright violations is to generate and embed an imperceptible signal (a watermark) in the original data, which can carry information, for example, about its owner.

According to its applications, watermarking methods can be for content authentication/tamper proofing or for copyright protection. In the first case, the goal is, for example, to detect changes made to the model. This requires what is called fragile or semi-fragile watermarking technologies, as presented in Boon-Lock et al. [24]: any small modification of the model greatly affects the watermark. For copyright protection the watermark should be perceptually invisible, statistically undetectable and resist a variety of modifications like rotation, translation, scaling or even simplification. For this application area a larger number of methods have been proposed, such as those of Benedens et al. [19], Ohbuchi et al. [99] and Zafeiriou et al. [165].

The watermark can be embedded in the spatial domain or in a transform embedding domain. In the spatial domain, watermarks are embedded by modifying the geometry [97],

topology, connectivity [57] or surface normals. These methods are not very complex but they are usually sensitive to several attacks. A recent example is the work presented by Bors [26].

In the transform domain, the spectral approach is very common. The decomposition of the target mesh into a spectral domain has been performed using a wide variety of methods like wavelet transforms [74], multi-resolution analysis [164], spectral mesh analysis [98] or, more recently, orthogonal basis functions [162]. Apart from the spectral domain approaches, other transform techniques were proposed. An example is a method by Song et al. [139] which transforms the mesh into an image and then embeds the watermark using image-based watermarking.

## 3.4 Evaluating and Comparing Mesh Features

The computation of mesh properties or features (e.g., curvature) is usually a first step towards polygonal mesh analysis and comparison. The comparison between two meshes (e.g., an original mesh and a processed mesh resulting from it) is then carried out by evaluating, for a given property, the distribution of differences between the two meshes.

Many methods have been presented in the literature for computing polygonal mesh properties, namely, for estimating curvature, saliency, and mesh element (e.g., triangles) quality. Other methods have been proposed for computing particular difference measures between two meshes: geometric distance, attribute deviation and visual differences.

### 3.4.1 Mesh Analysis

In what follows a short overview on some methods to compute polygonal mesh properties is presented.

#### Mesh Curvature

There are many methods for estimating surface curvature on polygonal meshes, the following being widely used: the quadric fitting method of Hamann [56]; Taubin's algorithms [147] with later improvements proposed by Surazhsky et al. [145]; the discrete Gaussian and mean curvature estimation methods of Meyer et al. [94]; a cubic-order algorithm for approximating principal direction vectors proposed by Goldfeather et al. [50]; the normal based estimation of the curvature tensor described by Thiesel et al. [152]; and the method for estimation of curvature and its derivatives of Rusinkiewicz [124]. Recently new estimation methods have appeared in the literature: see Agam et al. [11] and Razdan et al. [110]; the latter uses biquadric Bézier patches.

A comparison of Gaussian and mean curvature estimation methods can be found in Magid et al. [93].

Figure 3.17: On the left, the *Stanford Armadillo* and on the right its mesh saliency distribution [84].

### Mesh Saliency

A purely curvature-based metric might not produce correct information about the perceptual importance of a mesh feature. For example, a high curvature feature in the middle of a flat region will probably be perceived as important; but a flat region in the middle of densely repeated high curvature bumps will also be perceived as important. Mesh saliency is a measure of such regional importance [84] by identifying the mesh surface areas which are judged to be of greater interest, i.e., which present a larger number of features.

Over the years this concept has been used in order to enhance simplification methods: Kho et al. [77] and Pojar et al. [108] determined salient features by user selection. Yee et al. [163] computed the saliency of 3D models based on their 2D projections. In the recent work of Howlet et al. [65] saliency is ascertained using an eye-tracking device, and it is shown that using this data during the mesh simplification process can help preserve visual fidelity. A few authors compute saliency directly from a 3D model: Watanabe et al. [158] detect salient curvature features; Hisada et al. [60] detect perceptually salient features using a 3D skeleton. Recently, Lee et al. [84] presented a method which allows the computation of mesh saliency based on mean curvature information. They point out that a saliency measure must be computed for different scales, since a feature that is important at a particular scale might not remain so at a different one.

### Mesh Element Quality

When assessing quality, it is important to evaluate the quality of each one of the elements that compose a mesh. Several metrics have been proposed in order to assess this (particularly for triangles) based on minimum angle, maximum angle, edge ratio, edge to inradius, etc. Pébay et al. [104] analysed several triangle quality measures, studied their extremal properties and examined their asymptotic behavior, in order to better understand the information they might provide.

More complex element analysis has also been proposed. Recent approaches are those

of Huang [66], where three mesh quality measures characterizing the shape, alignment and adaptation features of elements are introduced, and of Cao [32] where the relation between triangle geometric properties and the associated error of linear interpolation is analysed.

#### Smoothness Analysis

Mesh smoothness is a good measure of the "visual quality" of a mesh [75]. It is possible to measure the smoothness of a mesh by computing the difference between a vertex position and the position of the centroid of its direct neighbors: mesh smoothness is larger when this value is smaller.

Another method used to analyse smoothness is through the visualization of isophotes or reflection lines [27]. Isophotes are lines of constant illumination on a surface. To compute these lines, a Lambertian surface [13] is considered (i.e., with purely diffuse reflection). This means the isophotes are independent of the viewpoint. The resulting images allow the detection of surface irregularity when compared with standard shading. The user can examine the lines, rate their smoothness and transfer this information to the surface: $C^k$ continuous surfaces lead to $C^{k-1}$ continuous isophotes.

Figure 3.18: Reflection lines on $C^0$, $C^1$ and $C^2$ surfaces [27].

Reflection lines (see Fig. 3.18) are rendered assuming a specular surface (thus, they depend on the viewpoint) and they have been traditionally used in the automotive industry to evaluate car surface quality by using an arrangement of parallel fluorescent lights placed above the car. Just like the isophotes, reflection lines allow obtaining information about surface continuity: if a surface is $C^k$ continuous then the reflection lines will be $C^{k-1}$ continuous.

### 3.4.2 Mesh Comparison

In what follows a short overview on methods used to compare polygonal meshes is provided.

#### Geometric Distance

Geometric distance gives an idea of how close the shape of one surface is to the other. Several methods are proposed in the literature which allow measuring the distance between two meshes. An important aspect is surface sampling: it is usually required since, when comparing two meshes, they might have a different number of vertices (e.g., due to simplification)

or their vertices might not have a clear correspondence. Thus, in order to compare a vertex position with its correspondent on another mesh, the latter mesh surface must be sampled in order to find commmon comparison points.

Cignoni et al. [37] presented a method for measuring the geometric distance using surface sampling; similar approaches were presented by Aspert et al. [14], which focus on the Hausdorff distance, and by Roy et al. [123]. Recently, Guthe et al. [55] presented a method for fast and accurate computation of the Hausdorff distance.

Other approaches have also been presented: Inagaki et al. [67] described a method which uses pixel based search; Park et al. [101] presented a shape dissimilarity measure which uses the depth buffer and a *surface roving* method, in order to measure distances between corresponding points on different 3D models.

### Attributes Deviation

Attributes are data defined at each mesh vertex, e.g., color, normal and texture coordinates. These attributes can play an important role in models aimed for rendering. For example, since normals are used in lighting calculations, a large normal deviation might entail visual artifacts or significant differences when the mesh is rendered.

Cohen et al. [41] described a method to compute texture deviation between two meshes. Recently, Roy et al. [123] proposed a generic attribute deviation metric which allows the computation of the deviation for several types of attributes.

### Visual Metric

In order to capture the visual difference between an original model and its approximation, Karni et al. [75] proposed a visual metric based on the geometric and Laplacian differences between two meshes, which is used in the context of quality assessment of a compression algorithm. The geometric difference is related with "physical" (coordinates) differences, while the Laplacian is related to visual/perceptual differences. Computing the visual metric is only possible for meshes having the same number of vertices, and a correspondence between them must be possible. A processing method which satisfies this requirement is, for example, smoothing.

More recently, Sorkine et al. [141] used this visual metric and proposed assigning a higher weight to the Laplacian difference (smoothness), since it is deemed more important to the perceived visual quality.

### Mixed Measure

Following the results presented in Sousa Santos et al. [143], which suggest that (1) the Geometric Distance is a better estimator of perceived quality for strongly simplified models, and that (2) the Normal Deviation is a better estimator of perceived quality for less simplified models, a mixed measure can be proposed which weights those two deviations according to the simplification level of a model.

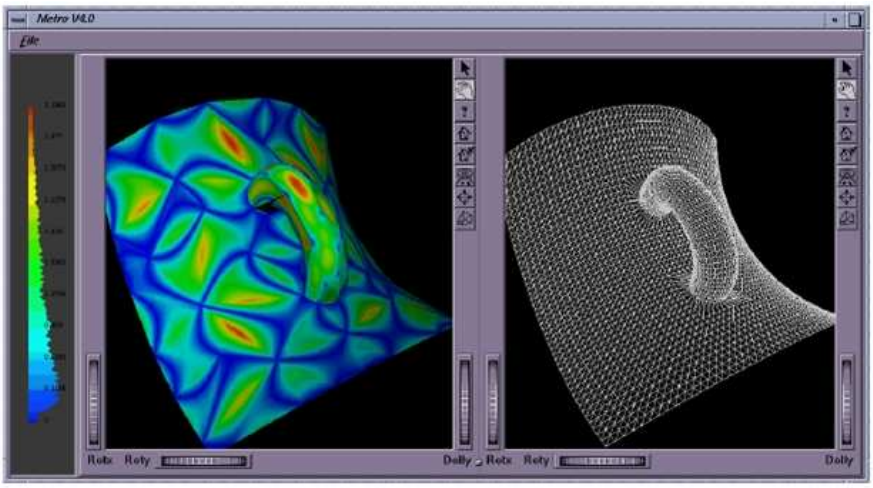


Figure 3.19: The Metro graphical output window [37].

Such mixed measure is computed as $C(v_i) = (1.0 - \alpha)G(v_i) + \alpha N(v_i)$, where $G(v_i)$ and $N(v_i)$ denote the Geometric and Normal Deviations for vertex $v_i$ and $\alpha = m/M$, where $m$ is the number of vertices of the processed mesh and $M$ the number of vertices of the original (reference) mesh. The latter is still a very crude approximation of an ideal blending factor and further research is clearly needed here.

## 3.5 Mesh Comparison Tools

Mesh comparison is usually carried out with the help of dedicated software tools providing the user with numerical data (e.g., minimum, mean and maximum difference values) and visual information (e.g., coloring a model according to the difference values measured at each vertex), and allowing the user to choose among several difference measures.

A few such tools, which allow mesh feature evaluation and the comparison of polygonal meshes, are described in the literature and an analysis of their features is presented next.

### 3.5.1 Metro

*Metro* is a command line tool, developed by Cignoni et al. [37] for Windows and Linux which can be found at `http://vcg.sourceforge.net/tiki-index.php?page=Metro`. An early Graphical User Interface, which was not available in the tested version, is presented in Fig. 3.19.

#### Main Features

*Metro* provides different methods for surface sampling (Monte Carlo sampling, subdivision sampling and similar triangles sampling).

It allows both numerical and visual mesh comparison. Among the numerical values provided is data about the input mesh characteristics (number of vertices and faces, surface area,

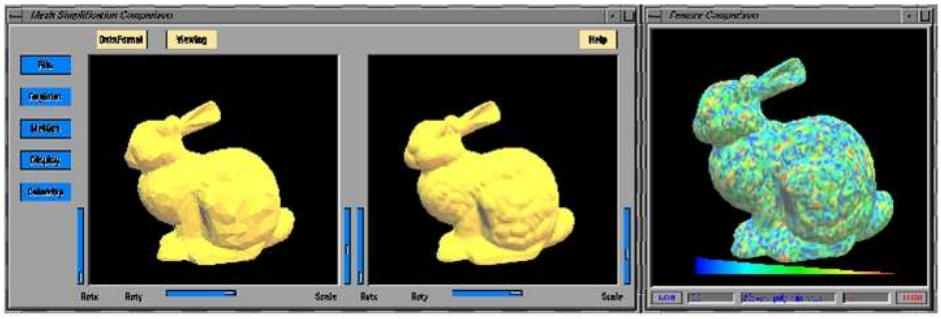


Figure 3.20: Graphical User Interface of the tool developed by Zhou et al. showing (from left to right) a processed model, the original model, and a model colored according to the difference between the two [166].

Figure 3.21: Several visual mapping options provided by the tool developed by Zhou et al. From left to right: overlay, rainbow mapping, white-black-white pseudo-coloring, glyph (high-pass filtered) and glyph (low-pass filtered) [166].

mesh volume, etc.), the minimum and maximum distances between two given meshes and their difference in volume. It is also possible to view a model colored according to the results obtained. Also provided is the computation of the Hausdorff distance.

#### Applications

*Metro* is the most used tool to assess differences between polygonal meshes. Some examples are the work of Cignoni et al. [36], where it has been used to compare among mesh simplification algorithms; Valette et al. [155], where it was used to assess the quality provided by a multiresolution scheme; and Müller et al. [96], where it was used to evaluate a mesh compression method.

### 3.5.2 Metrics and Visualization Tools for Surface Mesh Comparison

This is a tool developed for SGI workstations by Zhou et al. [166].

#### Main Features

This tool introduces some additional measures (namely, discrete surface curvature) and care was taken to present several visualization techniques, including side-by-side viewing of the compared models and results (Fig. 3.20), box-glyphs and animations (Fig. 3.21).

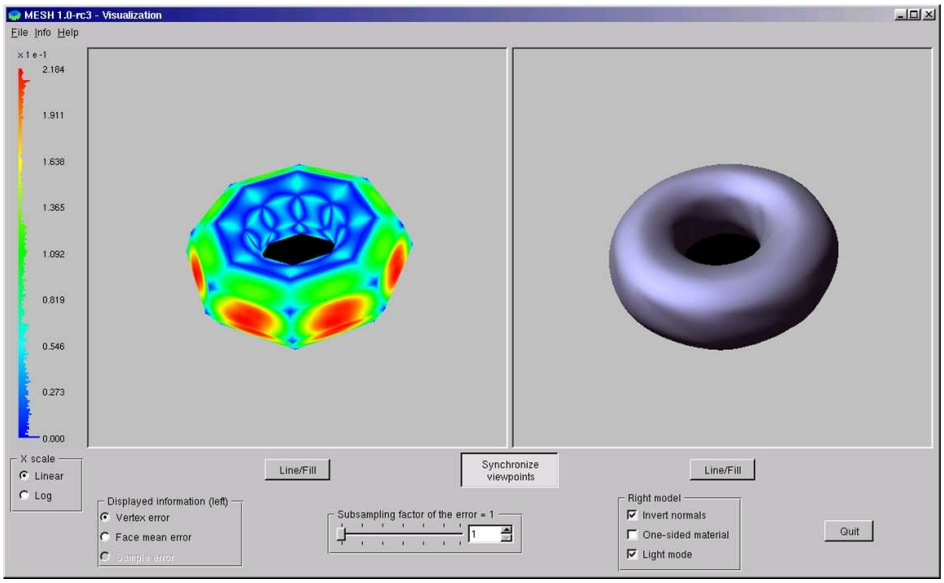



Figure 3.22: *M.E.S.H.* visualization window showing vertex error information for a torus model [14].

### Applications

The only application of this tool in the literature, to the best of my knowledge, is the one proposed in Zhou et al. [166] to compare different resolution meshes and meshes obtained with different methods.

The authors who cite it do it mainly because they use a discrete curvature computation method also proposed in the above mentioned paper.

## 3.5.3 M.E.S.H.

*M.E.S.H.* (Measuring Error between Surfaces using the Hausdorff distance) was developed by Aspert et al. [14], and can be found at `http://mesh.berlios.de/`. It can be run in both Windows and Linux systems.

### Main Features

*M.E.S.H.* uses the Hausdorff distance to measure the difference between two mesh models. It provides several numerical values, namely the main features of the input meshes and the minimum, mean, maximum and root-mean-square values of the results. A simple Graphical User Interface is provided to support results viewing and analysis which allows viewing a model colored according to the obtained results. The provided visualization window (Fig. 3.22) also allows synchronizing both original and colored model, and selecting some rendering options like wireframe.

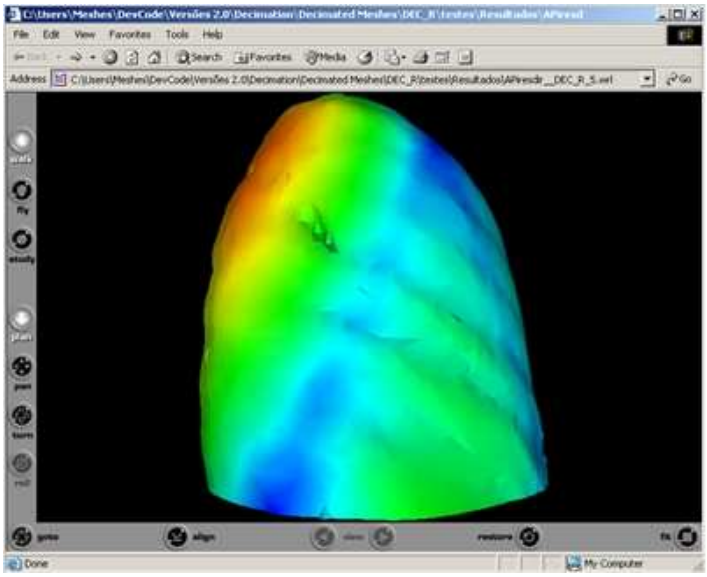


Figure 3.23: Internet Explorer window, using the Cortona plugin [1], showing a colored model returned by *MeshDev*, depicting the data obtained in the comparison of two polygonal models.

#### Applications

M.E.S.H. has been used in a wide variety of applications. As an example one can refer the work of Rajamani et al. [109], where it is used for quality assessment of a model for surgical visualization; Cohen-Steiner [42], where it is used to compute deviations for a new shape approximation method; Payan et al. [103], where it is used to assess the results obtained with a normal mesh compression method; and van Kaick et al. [156], where different metrics for mesh simplification are compared.

### 3.5.4 MeshDev

Finally, Roy et al. [123] presented a tool called *MeshDev*, which can be obtained at `http://meshdev.sourceforge.net/`.

#### Main Feaures

*MeshDev* allows the computation of geometric, normal and other attribute deviations such as color or texture. Similarly to the other tools, it provides several numerical values characterizing the input meshes and the obtained results. It is also possible to view models colored according to the computed deviations using a rainbow color scale. The colored models are returned in VRML format and need an external viewer to be analysed (see Fig. 3.23).

#### Applications

In Sousa Santos et al. [143], *MeshDev* is used to assess the quality of polygonal models obtained using different simplification levels and methods. The obtained results were then compared with perceived quality data obtained through an observer study; in Ho et al. [61], *MeshDev*

was used to assess the quality of simplified polygonal models obtained using a user-assisted mesh simplification framework.

## 3.6 Conclusion

This chapter provided a brief overview on Geometric Modeling using polygonal meshes focusing on the most popular mesh processing methods, and on the evaluation and comparison of mesh features, describing several methods and tools used for that purpose.

Polgonal meshes are used in many application domains and may be processed using various methods and with several particular goals. These mesh processing operations entail changes in the original meshes and the obtained results must be assessed in order to understand how they affect the meshes. It is important to quantify the amount of change introduced thus allowing the definition of confidence levels or selection of suitable algorithms.

Feature evaluation of polygonal meshes has been a subject widely explored in the literature and many computational measures exist which allow obtaining, for example, surface curvature. In general, these measures are available in software tools for mesh analysis and comparison, ranging from console applications to simple Graphical User Interfaces, which have allowed users to perform mesh quality assessment in numerous situations. But these tools lack some important features which would allow a more systematic usage. The following chapter will identify some of those missing features, and present a new mesh analysis and comparison tool which provides them.

# Chapter 4

# PolyMeCo

In Chapter 3 several mesh processing methods have been described and many others exist, such as remeshing or feature enhancement [38]. All these processing methods result in polygonal meshes which exhibit differences towards the original. As there are several ways of accomplishing the same mesh processing operation (e.g., compression), each one leading to different results depending on the algorithms used, it is important to measure these differences in order to understand which is the best algorithm and if the resulting differences are acceptable for a particular goal.

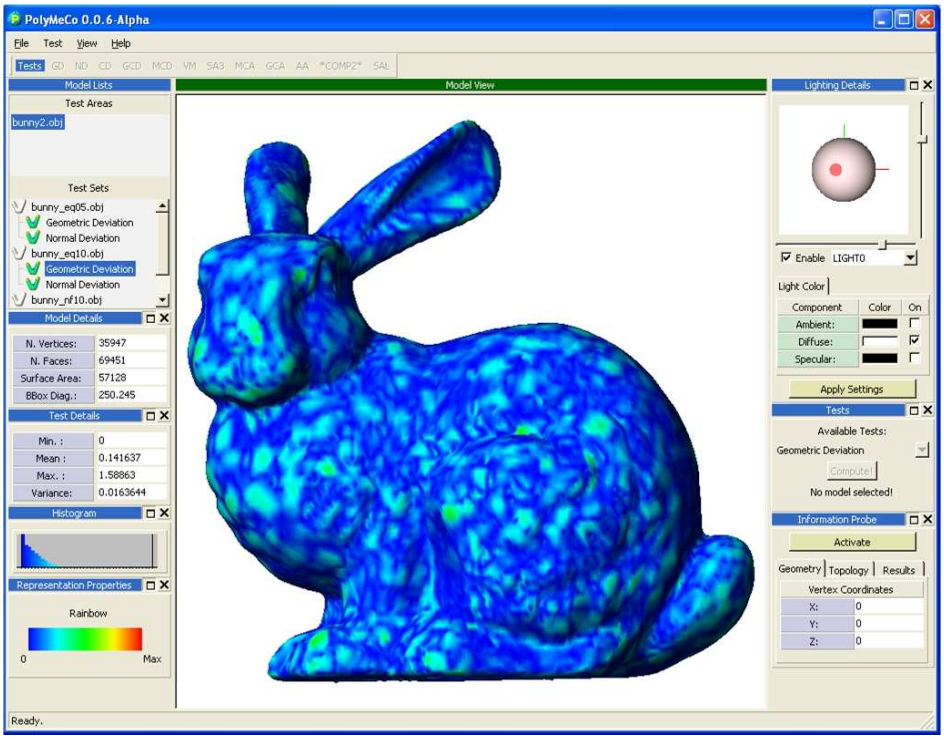



Figure 4.1: A view of PolyMeCo's user interface, while performing mesh comparison.

Such an assessment is essential for researchers developing new methods, as they have to fine tune them in order to provide better solutions than those already available, but also for users applying these methods for particular purposes. The users' role is of paramount importance,

since they often test or apply the mesh processing methods in unexpected situations which were not considered by the developers, revealing important results.

In order to help users to insert an assessment stage in their work flow, it is important that the tools available for that purpose provide all the necessary features to allow systematic assessment and enhanced insight on the obtained data.

PolyMeCo (**Poly**gonal **Me**sh **Co**mparison)[1] is such a tool, providing an integrated environment for mesh analysis and comparison (Fig. 4.1).

This chapter describes PolyMeCo's architecture and features. It starts by identifying the main features which PolyMeCo provides. Then, it presents a possible mesh analysis and comparison pipeline and, based upon it, describes PolyMeCo's main modules by stating their purpose and contents.

## 4.1 Desired Functionalities

The analysis of the comparison tools mentioned in Section 3.5 revealed the lack of several features which seem to be important, in particular if one aims to provide a tool for systematic use by researchers and non-researchers.

When dealing with polygonal models it must be possible to:

- Load several models during the same work session in a clear and systematic way;
- Compute different mesh features/properties and difference measures;
- Compute a mesh property or difference measure (e.g., geometric distance) for several models at once;
- Save the contents of the work session in order to allow resuming the work.

When dealing with the data obtained by computing a mesh property or difference measure for a particular model it must be possible to:

- Choose several ways of presenting the computed data;
- Visualize the data obtained using several computational measures for the same model;
- Compare data values obtained with the same computational measure for different (and comparable) models;
- Access computed values associated with a particular mesh entity (e.g., a vertex);
- Export the computed data in order to allow further analysis with statistical tools.

These features are all supported by PolyMeCo and will be described in greater detail ahead on this chapter.

[1] A first version of PolyMeCo was presented by the author in Silva et al. [135].

## 4.2 Polygonal Mesh Storage and GUI Development

After defining the desired features for PolyMeCo[2] it was important to choose which would be the data structure used to store and manipulate the polygonal models, and how would the Graphical User Interface (GUI) be developed. The development was to be performed in C++ using (primarily) *Microsoft Visual Studio*, in Windows environments, and *Eclipse* in Linux environments. Model rendering would be executed using *OpenGL* [131]. In what follows the chosen options are presented.

### 4.2.1 Data Structure

After analysing the alternatives described in Section 3.1.2, the OpenMesh [28] library was chosen to provide the data structure for mesh storage and manipulation, given its simplicity, its mesh manipulation tools (e.g., vertex, face and one-ring iterators) and the purpose of its authors in expanding it with additional features.

Although this library is developed in *Linux* environments, there was no major problem in porting it to use in *Microsoft Visual Studio 2005*.

### 4.2.2 Graphical User Interface

One of the main goals was to develop PolyMeCo as a multi-platform tool which could be used either in *Microsoft Windows* (the primary development environment) or *Linux* environments. There are several open source libraries available which allow Graphical User Interface (GUI) development, independently of the platform. Some examples are:

- **FOX Toolkit**

  FOX Toolkit [3] is a C++ based toolkit for developing Graphical User Interfaces which provides several controls and features, such as drag and drop, and *OpenGL* widgets for 3D graphical manipulation. Many of the provided widgets can be created using just one line of code and it is possible to create new widgets by deriving classes from existing ones. Interfaces developed with Fox Toolkit look the same regardless of the operating system.

  FOX Toolkit can be used on several operating systems including Linux, IRIX, HP-UX and Microsoft Windows.

- **Qt**

  Qt [8] is a C++ based library containing several modules which provide, beyond GUIs development, database, networking and multi-threading features. It also provides classes for working with OpenGL 3D graphics. Interfaces developed with Qt always present a native look and feel depending on the operating system.

  Qt can be used to develop applications targeting several operating systems like Linux, IRIX, HP-UX, Mac OS and Microsoft Windows.

[2] A version for test purposes can be obtained in `http://www.ieeta.pt/polymeco`

- **wxWidgets**

  wxWidgets [10] is a library developed in C++, which can be used to develop GUIs from languages as C++, Python, Perl and C#. It supports several controls and features including OpenGL integration, database, network and multi-threading support.

  Similar to what happens with Qt, GUIs developed using wxWidgets present a native look and feel according to the operating system.

  wxWidgets can be used to develop GUIs for Linux, Mac OS and Microsoft Windows.

After considering the above possibilities and aiming for simplicity, flexibility and similar look and feel across plataforms, Fox Toolkit was chosen.

## 4.3 Mesh Analysis and Comparison Pipeline

The process of analysing and comparing meshes can be described by the pipeline presented in Fig. 4.2. After loading polygonal models, some of their features are described by computational measures. Then, the obtained results are mapped to a suitable representation. After this mapping has been performed, the chosen representations are presented to the user who may interact with them and change parameters along the pipeline in order to, for example, choose another computational measure, or the representation mapping used.

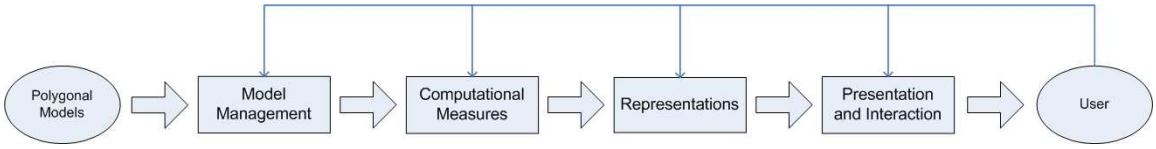



Figure 4.2: Mesh analysis/comparison pipeline.

Notice that this pipeline is analogous to the Visualization pipeline presented in Chapter 2. The *Computational Measures* stage comprises obtaining raw data and transforming it; the *Representations* stage comprises building *Visual Structures* and mapping data onto them; and the *Presentation and Interaction* stage comprises all tasks regarding interacting with the visualization.

By taking into account the proposed pipeline, PolyMeCo should include four main functional blocks:

1. *Model Management*
2. *Computational Measures*
3. *Representations*
4. *Presentation and Interaction*

In the following sections each one of those blocks will be described in more detail along with the features available in PolyMeCo.

## 4.4 Model Management

In the mesh comparison process it is necessary to distinguish between the original (reference) mesh and those which are its versions obtained using some mesh processing operations. In PolyMeCo, loading an original model creates a new *Test Area*. Each *Test Area* can contain multiple model *Test Sets*. A *Test Set* is created when a (processed) version of the original model is loaded for comparison. Finally, each *Test Set* will contain the data for all the measures computed for the model version it refers to. Figure 4.3 shows this hierarchy.

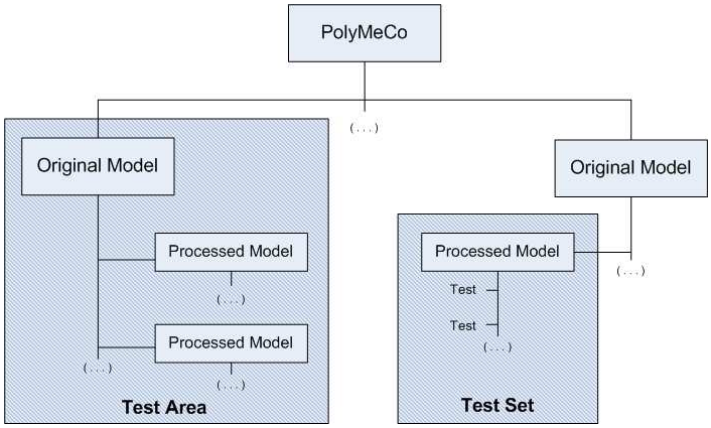



Figure 4.3: Diagram depicting the difference between *Test Areas* and *Test Sets*.

## 4.5 Computational Measures

*Computational Measures* allow mesh analysis and comparison by providing information regarding mesh properties (e.g., curvature) or deviations between meshes (e.g., geometric distance).

A *Computational Measure* (see Fig. 4.4) uses information regarding the currently active models (i.e., the selected *Test Area* and *Test Set*). Then, depending if it is an analysis or a comparison, it uses information of one or both to compute the results.

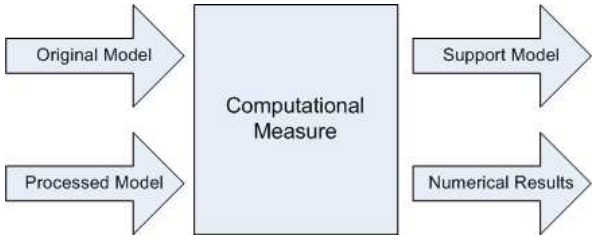



Figure 4.4: Diagram for a computational measure.

It then provides the numerical values obtained and a 3D model which may be used to support the data representation. This model can be the original model, in a comparison, or the processed model, when computing an analysis measure. There are also situations where

this model can be different, e.g., when the computed measure has no direct connection with the geometry of the models.

In what follows some details are provided concerning the computational measures currently included in PolyMeCo.

### 4.5.1 Intrinsic Properties

Intrinsic properties[3] allow the measurement of a particular property of a mesh. In what follows, the intrinsic properties available in PolyMeCo are described[4].

- **Smoothness** – This measure gives an idea of the smoothness of a model surface. For each vertex, the average position (centroid) of its direct neighbors (one-ring vertices) is obtained and the distance to the vertex is computed: when this distance is small, the region to which it corresponds is smoother. Equation 4.1 is used to compute the smoothness vector for each vertex $v_i$, where $N_1(v_i)$ denotes the set of direct neighbors of vertex $v_i$:

$$S(v_i) = v_i - \frac{1}{\#N_1(v_i)} \sum_{v \in N_1(v_i)} v \tag{4.1}$$

  The distance can then be computed by taking the norm of $S(v_i)$.

- **Mean and Gaussian Curvatures** – It is possible to compute the Gaussian and mean curvatures for each vertex of a mesh. The algorithms implemented in PolyMeCo follow the ones described by Meyer et al. [94].

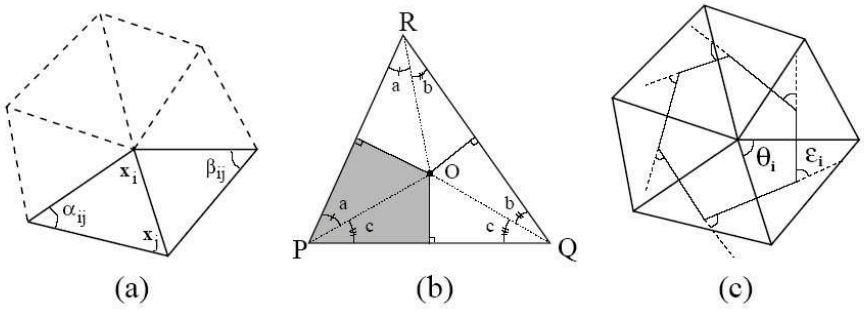



Figure 4.5: (a) Direct neighbors and angles opposite to an edge; (b) Voronoi region on a non-obtuse triangle; (c) External angles of a Voronoi region [94].

  The value of the mean curvature $\overline{\kappa}$, for vertex $v_i$, is obtained by taking half of the norm of the vector defined in Eq. 4.2, where $N_1(v_i)$ again denotes the set of direct neighbors of vertex $v_i$; $\alpha_j$ and $\beta_j$ are defined as shown on Fig. 4.5.

$$\overline{\kappa}(v_i) = \frac{1}{2A_{mixed}} \sum_{v_j \in N_1(v_i)} (\cot \alpha_j + \cot \beta_j)(v_j - v_i) \tag{4.2}$$

[3]Here used in the sense of a property which is directly computed from a model data, irrespective of any other reference model; not in the Differential Geometry sense as in "intrinsic curvature".

[4]Whenever applicable, the notation used is that of the article where the property has been presented.

The value of the Gaussian curvature $\kappa_G$, for vertex $v_i$, is obtained using Eq. 4.3, where $\theta_j$ is the angle of face $j$ at vertex $v_i$.

$$\kappa_G(v_i) = (2\pi - \sum_{j=1}^{\#f} \theta_j)/A_{mixed} \tag{4.3}$$

The scalar $A_{mixed}$, used in Eqs. 4.2 and 4.3, represents an area: if there is no obtuse triangle in the one-ring neighborhood of a vertex, it is obtained using the Voronoi area defined by Eq. 4.4; whenever there is an obtuse triangle, the sum term associated with it in Eq. 4.4 is replaced by the value of one quarter or half of the triangle area according to the value of the angle on vertex $v_i$.

$$A_{Voronoi}(v_i) = \frac{1}{8} \sum_{v_j \in N_1(v_i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \left\| v_i - v_j \right\|^2 \tag{4.4}$$

- **Saliency** – This measure provides an analysis of regional importance [84] by identifying the mesh surface areas which are judged to be of greater interest, i.e., which present a larger number of features. As stated in [84], it must not be forgotten that this kind of measure must be computed for different scales, since a feature that is important at a particular scale may not remain so at a different one.

  To obtain the saliency values for a mesh, a Gaussian-weighted average is first computed for each vertex using Eq. 4.5, where $\mathscr{C}(x)$ denotes the mean curvature for vertex $x$ and $N(v, 2\sigma)$ denotes the set of neighbors of vertex $v$ in a neighborhood of $2\sigma$ radius.

$$G(\mathscr{C}(v), \sigma) = \frac{\sum_{x \in N(v,2\sigma)} \mathscr{C}(x) exp\left[ - \left\| x - v \right\|^2 / (2\sigma^2) \right]}{\sum_{x \in N(v,2\sigma)} exp\left[ - \left\| x - v \right\|^2 / (2\sigma^2) \right]} \tag{4.5}$$

  The saliency for a vertex can then be computed using Eq. 4.6. To compute saliency values for different scales one can use Eq. 4.6 and vary the value of $\sigma$ (i.e., the neighborhood radius).

$$\mathscr{S}(v) = |G(\mathscr{C}(v), \sigma) - G(\mathscr{C}(v), 2\sigma)| \tag{4.6}$$

- **Triangle Quality** – Triangle quality is measured by computing the value of the minimum angle for each mesh triangle, which provides a general idea of how triangles are shaped along a mesh. When the minimum angle is close to 60 degrees, the triangle is close to equilateral; when the minimum angle is small it is not possible to say whether it belongs to a needle or a flat triangle.

  For a more accurate analysis of triangle quality there are several other measures, which are studied in [104] and which we intend, in a short term, to add to PolyMeCo.

### 4.5.2 Difference Measures

Difference measures allow the comparison of properties between meshes. The following difference measures are available in PolyMeCo:

- **Geometric Distance** – It is possible to measure the distance (geometric difference) between each mesh vertex and its corresponding position on another mesh. When comparing two meshes, it is usual that they have a different number of vertices (e.g., due to simplification) or that their vertices do not have a clear correspondence. Thus, in order to compare a vertex's position with its correspondent on another mesh, the latter mesh's surface must be sampled in order to find common comparison points. Due to this particularity, the implementation of this measure available in PolyMeCo is largely based on the one furnished by MeshDev [123], which already provides a mesh surface sampling method.

- **Normal Deviation** – It is possible to measure the difference between the normal to each face of a mesh and its corresponding face on another mesh. Due to the need for sampling the surface of the latter, in order to find common comparison points, the implementation of this measure provided in PolyMeCo is strongly based on the one available in MeshDev [123].

- **Mixed Measure** – Following the results presented by the author and his colleagues in Sousa Santos et al. [143] and Silva et al. [137], which suggest that (1) the Geometric Distance is a better estimator of perceived quality for strongly simplified models, and that (2) the Normal Deviation is a better estimator of perceived quality for less simplified models, a mixed measure is available in PolyMeCo which tries to weight those two measures according to the simplification level of a model.

  Such mixed measure is computed as $C(v_i) = (1.0 - \alpha)G(v_i) + \alpha N(v_i)$, where $G(v_i)$ and $N(v_i)$ denote the Geometric Distance and Normal Deviation for vertex $v_i$ and $\alpha = m/M$, where $m$ is the number of vertices of the processed mesh and $M$ the number of vertices of the original (reference) mesh. The latter is still a very crude approximation of an ideal blending factor and further research is clearly needed here.

- **Mean and Gaussian Curvature Deviations** – The curvature deviation measures available in PolyMeCo use the already described methods in order to compute the curvature for each mesh model, and the surface sampling capabilities provided by MeshDev [123] to compare the curvature of the two meshes.

- **Visual Metric** – It gives a measure of the visual change that a model has suffered, based upon a *physical* distance and a Laplacian operator as described by Karni et al. [75]. The implementation available in PolyMeCo is based on the changes proposed by Sorkine et al. [141]. This metric can only be applied to compare between models where vertex correspondence is possible. This implies that both models must have the same number of vertices, referred by the same order (to provide vertex correspondence).

The *physical* distance between the models is obtained by computing the root-mean-square (RMS) value of the vertex distances as in Eq. 4.7, where $n$ is the number of vertices of the meshes and $Q(v_i)$ denotes the corresponding vertex of $v_i$ on the original mesh.

$$M_q = \left( \sum_{i=1}^{n} \|v_i - Q(v_i)\|^2 \right)^{1/2} \tag{4.7}$$

The geometric Laplacian for each vertex is computed using Eq. 4.8, where $l_{ij}$ is the Euclidean distance between $v_i$ and $v_j$ and $N_1(v_i)$ denotes the set of direct neighbors of vertex $v_i$. Then, using the values obtained in Eq. 4.8, the value of $S_q$ is computed (Eq. 4.9) and used to obtain the value of the visual metric (Eq. 4.10).

Sorkine et al. [141] argue that the parameter $\alpha$ on Eq. 4.10 needs to be smaller than the one proposed by Karni et al. ($\alpha = 0.5$), which seems to make sense since $S_q$ has a more significant visual effect. We have set $\alpha = 0.15$, but further research is needed in order to more clearly understand the impact of this parameter on the results obtained using this metric.

$$S(v_i) = v_i - \frac{\sum_{j \in N_1(v_i)} l_{ij}^{-1} v_j}{\sum_{j \in N_1(v_i)} l_{ij}^{-1}} \tag{4.8}$$

$$S_q = \left( \sum_{j=1}^{n} \|S(v_i) - S(Q(v_i))\|^2 \right)^{1/2} \tag{4.9}$$

$$E_{vis} = \alpha M_q + (1 - \alpha) S_q \tag{4.10}$$

## 4.6 Representations

In Visualization, data is mapped onto representations, which augment a spatial substrate with markers and graphical properties to encode information [33].

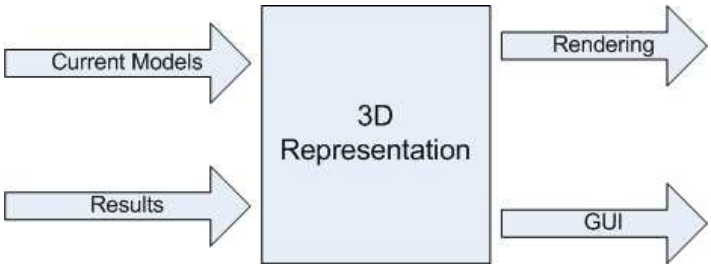



Figure 4.6: Diagram for a *Representation*.

In PolyMeCo a *Representation* provides a proper representation of data obtained with one of the available computational measures.

As inputs, a *Representation* (see Fig. 4.6) receives information regarding the current active models (i.e., the current *Test Area* model and, if selected, the current *Test Set* model and the current test). Associated with these models there can be information which will then be represented, e.g., the data values obtained with a computational measure.

As outputs, a *Representation* provides a proper rendering of the representation and a Graphical User Interface (GUI) template which can be used to provide representation details to the user (e.g., a caption describing what is being visualized), or to configure particular properties of the representation (e.g., transparency level).

All options regarding the properties of a *Representation* can be accessed by right-clicking with the mouse over it.

In what fallows a description of the *Representations* available in PolyMeCo is provided.

### 4.6.1 Numerical Values

The easiest way for providing the user with feedback about the computed measures is by presenting some numerical values that characterize them and/or the meshes for which they were computed. The following values are provided:

- **Mesh Features** – Number of vertices, number of faces, bounding box diagonal, surface area.
- **Statistical Data** – Minimum, mean and maximum values, and variance are provided for each computed measure.

### 4.6.2 Model Rendering

A simple way of enabling mesh analysis and comparison is by depicting the models and allowing the user to visually inspect their surfaces. To help on the analysis, it is possible to choose several rendering options (Fig. 4.7):

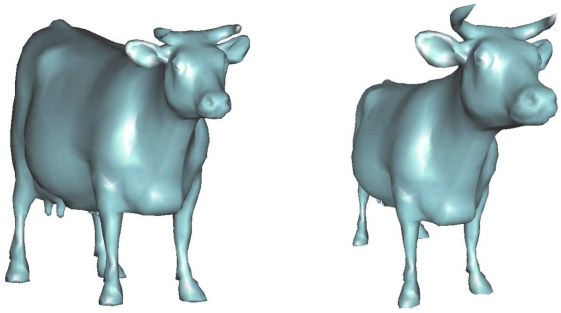
- **Projection**: orthogonal and perspective projections;
- **Pre-defined views**: front, back, top, bottom, left and right;
- **Element rendering**: vertices, edges (wireframe) and faces.

The user is also free to change the position, rotation and size of any model presented in PolyMeCo and, at any time, a *Reset* option is available which restores the initial viewing conditions (i.e., those which allow viewing the whole model centered on screen).

### 4.6.3 Colored Model

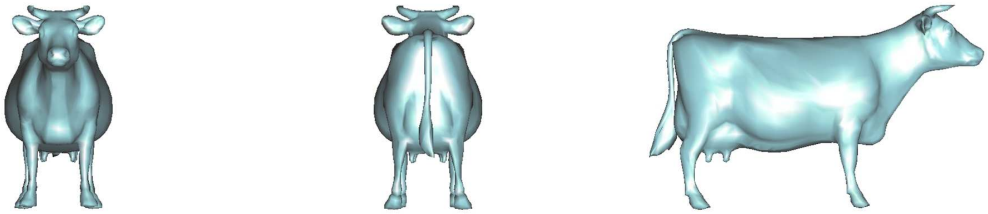
Through a colored model it is possible to show the user the distribution, over the mesh surface, of the data obtained with a particular computational measure: each vertex/face is colored

Projections

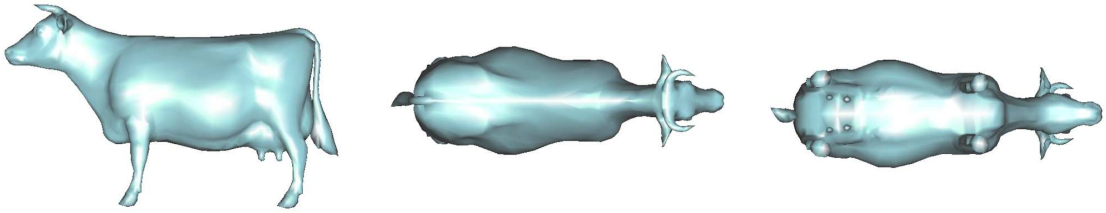


(a) Orthogonal projection (b) Perspective projection

Pre-defined views

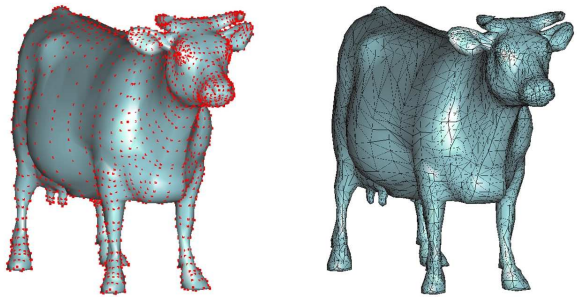


(c) Front (d) Back (e) Left



(f) Right (g) Top (h) Bottom

Element Rendering



(i) Vertices (j) Wireframe

Figure 4.7: Rendering options available in PolyMeCo.

according to the value obtained for each of them. This coloring can be done by mapping the values range to a particular color scale.

In Chapter 2 several color scales are described along with several guidelines to build and apply them to data sets. A usable color scale must respect a set of principles and be properly chosen in order to enhance the insight it provides on the data it refers to. It is also important to understand its disadvantages. An example is the rainbow color scale which is commonly used in many situations but is not, according to Rogowitz et al. [119] and, more recently, Borland et al. [25], the proper choice in many application scenarios. As a first approach towards a proper use of color in PolyMeCo several color scales described in the literature are provided, namely: the rainbow, greyscale, linearized greyscale, blue-to-cyan, blue-to-yellow, linearized optimal [85] and heated-object color scales.

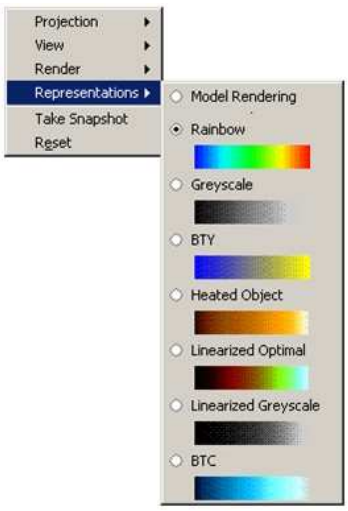



Figure 4.8: Menu presenting the color scales available in PolyMeCo

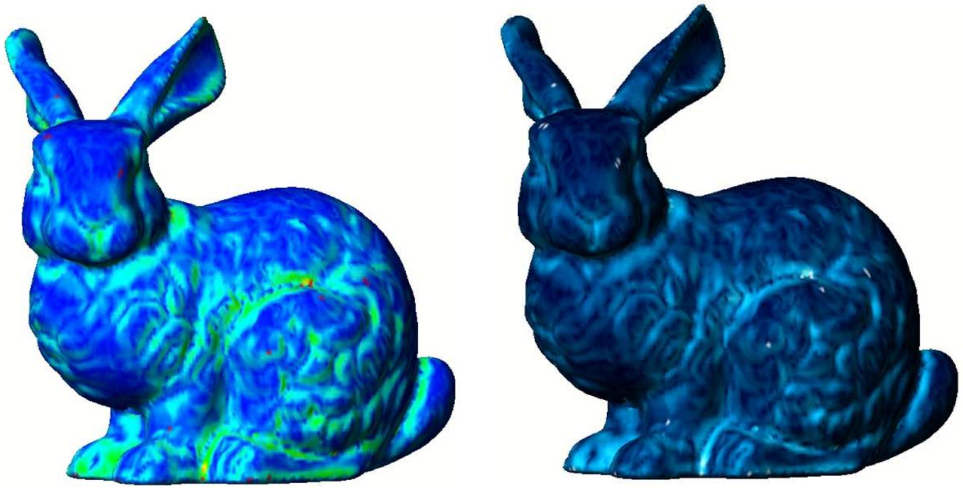
Currently, the user is allowed to choose, from this set of color scales (see Fig. 4.8), the one which provides the best visualization results. This is still a very naive approach to a correct use of color as it depends totally on user judgment, but provides some alternatives which he/she can explore.

On Fig. 4.9 (top row) computed results for the Normal Deviation *Computational Measure* are presented, using a model colored according to a rainbow and a blue to cyan color scales. Figure 4.9c shows part of a model colored according to the computed triangle quality.

It would be desirable that PolyMeCo somehow suggested the more appropriate color scale for each particular case based, among others, on the spectral features of the data and on the type of task the user wants to accomplish [20]. This would be an important step forward towards a more complete application of the principles described in Chapter 2.

### 4.6.4 Model Superposition

It is possible to view a reference model superimposed on one of its processed versions. A first choice is to render both models in a "solid" way, using different colors. This allows the



(a) (b)

(c)

Figure 4.9: Computed results are presented using a colored model: (a) Normal Deviation using a rainbow color scale and (b) a blue to cyan color scale ; (c) Triangle Quality using a rainbow color scale, to color the faces according to the minimum angle, and flat shading.

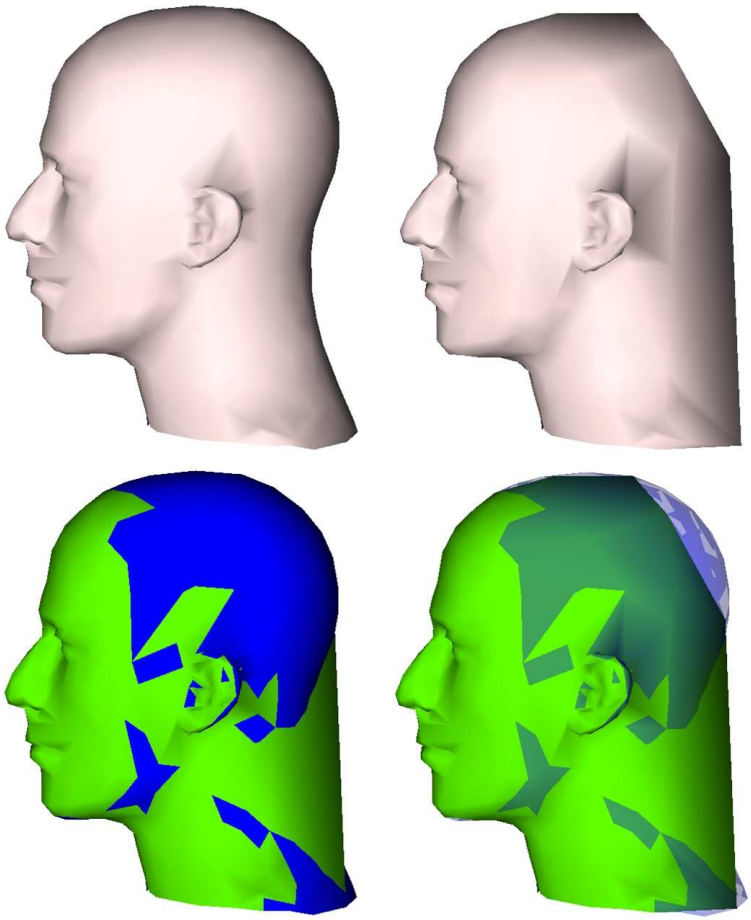


Figure 4.10: Model superposition. On top: the original model and the processed model; below: model superposition using solid rendering depicting a region where the orginal model overlaps the processed one, and model superposition with transparency applied to the original model, allowing volume comparison.

perception of the areas where, for example, the processed model overlaps the original model. A second choice is to render the original model with some degree of transparency, in order to let the user perceive the differences in volume between the two superimposed models. Figure 4.10 shows these two alternatives.

In addition, it is also possible to view the model superposition with both models rendered in wireframe.

### 4.6.5 Statistical Representations

Although model coloring gives a good idea of the distribution of the values obtained using a *Computational Measure* along a model's surface, showing where they occur, a global idea of the distribution characteristics can be difficult to obtain, due to the impossibility of viewing the whole model surface simultaneously. The provided statistical representations, described below, may help to better understand and compare distributions of computed measures.

- **Histograms** – Histograms provide information complementary to the one given by

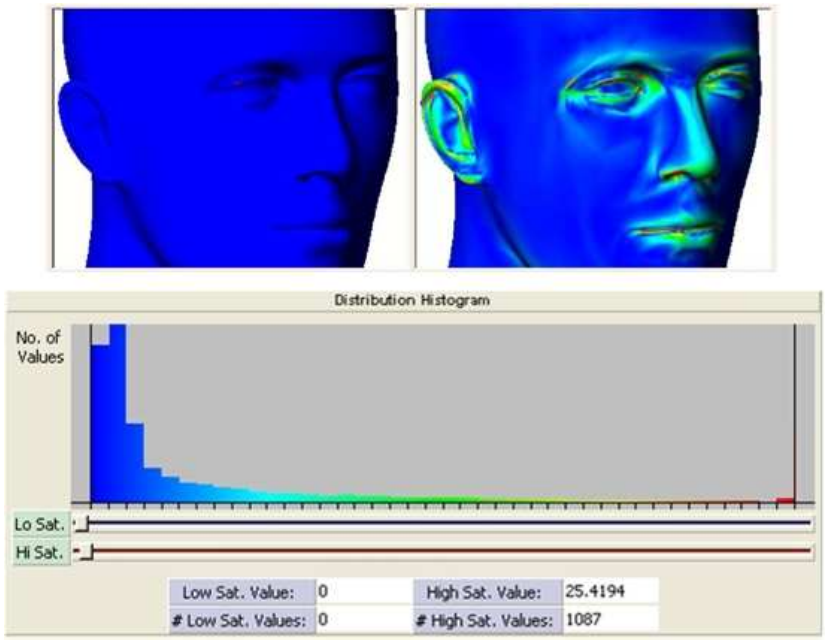



Figure 4.11: High saturation value definition in order to better visualize the representation of curvature over the mesh surface. Top, original colored model and colored model after high saturation value definition; bottom, the resulting histogram.

colored models: the user can, at a glance, get an idea of the global characteristics of the computed measure distribution.

A problem that can occur when assigning a range of values to a color scale is that of scale stretching: if outliers are present on the data, they may stretch the scale too much and not allow perceptible differences between close values. In order to deal with this issue, the histogram widget available in PolyMeCo provides a way of choosing the range of values that will be represented by the chosen color scale. By using a slider, the user can choose a high saturation value, i.e., all values above it will be represented using the "last color" of the color scale. A similar slider is available for defining a low saturation value, all values below it being represented by the "first color" of the color scale. The histogram will then use the first bar to represent the low saturated values and the last for the high saturated values, leaving others to represent the distribution of the remaining (in-between) values.

Figure 4.11 shows a model colored according to its mean curvature. Due to the presence of outliers on the data, it is not possible to visualize the curvature distribution. By defining a high saturation value using the slider provided on the histogram widget, it is possible to better perceive the curvature distribution. Note that the histogram bars are also colored with the chosen color scale.

This feature can also be used to highlight values in a colored model. Figure 4.12 shows, on top, several models colored using several color scales, depicting results obtained with the Geometric Distance *Computational Measure*. On bottom, the same models

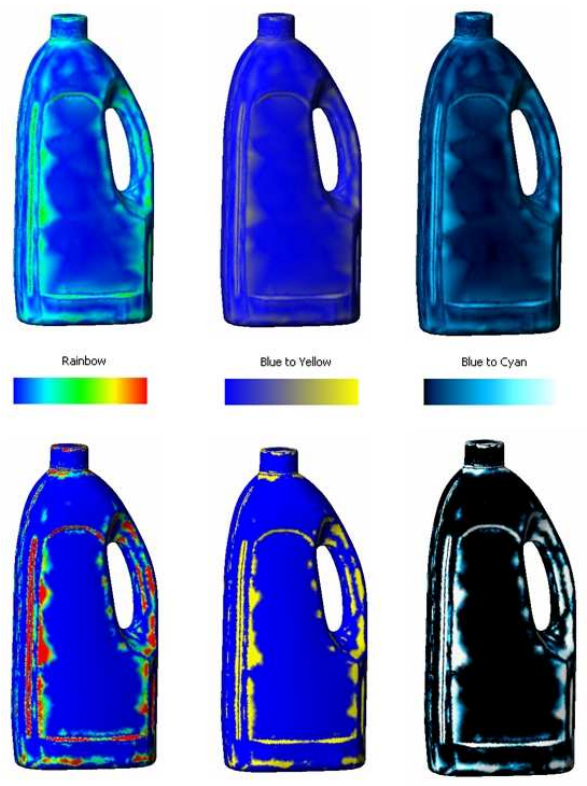



Figure 4.12: Top, models colored using several color scales, depicting the results obtained by computing the Geometric Distance. Bottom, same models but now with the color mapping modified in order to highlight the surface regions with larger associated values.

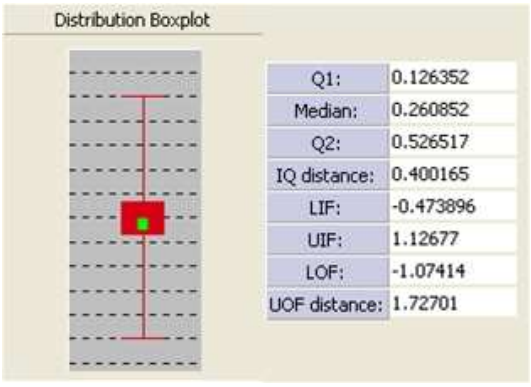



Figure 4.13: Boxplot and some data characterizing it.

are presented but the color mapping has been modified by adjusting the high saturation level. Notice how the surface regions with larger values associated are clearer to identify.

- **Boxplots** – Boxplots are very useful when comparing data sets, since they allow comparing and analysing the symmetries and ranges of the data, and detect the presence of outliers; thus, they can help setting the saturation values on the histogram widget. Figure 4.13 presents a boxplot drawn for a particular data set.

## 4.7 Visualization Modes

Concerning the *Presentation and Interaction* functional block, the first components are *Visualization Modes*.

A *Visualization Mode* (see Fig. 4.14) takes as inputs information regarding the current active models or even a list of models selected from those already loaded. It then uses the

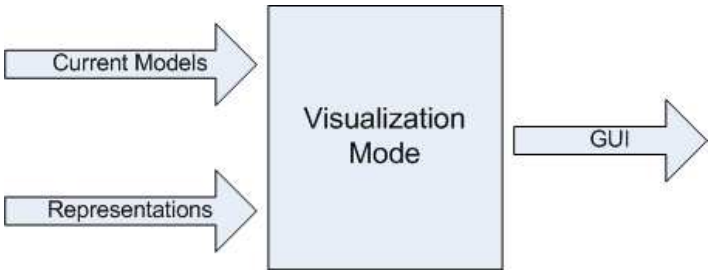



Figure 4.14: Diagram for a *Visualization Mode*.

available *Representations* to compose a Graphical User Interface which allows the user to gain a greater insight into the information he/she is visualizing. It can simply show one of the available *Representations*, several (coordinated) *Representations* for the same data, or provide ways of comparing among several results by using the same *Representation* for all of them in a side-by-side view.

Next, a description of all *Visualization Modes* available in PolyMeCo is provided.

### 4.7.1 Original vs Processed vs Colored Model

In this *Visualization Mode* the user is presented with the original model, the processed model and a colored representation of the data (e.g., deviation values) obtained using the selected *Computational Measure*. This allows the visual comparison of the two models, for example, to quickly identify areas where computed values are higher/lower, and then perform a more detailed analysis.

It is possible to manipulate any of the models and change its position, orientation and size. This manipulation has the particularity of maintaining all models synchronized, i.e., a change in position, orientation or size in one of the models is also applied to the others. Figure 4.15 illustrates this *Visualization Mode*.

### 4.7.2 Extended Results Viewing

This *Visualization Mode* allows the simultaneous presentation of a colored model, a histogram and a boxplot depicting the results obtained with one *Computational Measure*. In this mode it is possible (as described earlier) to define high and low saturation values in order to enhance the way results are presented using the colored model. Figure 4.16 shows this *Visualization Mode*.

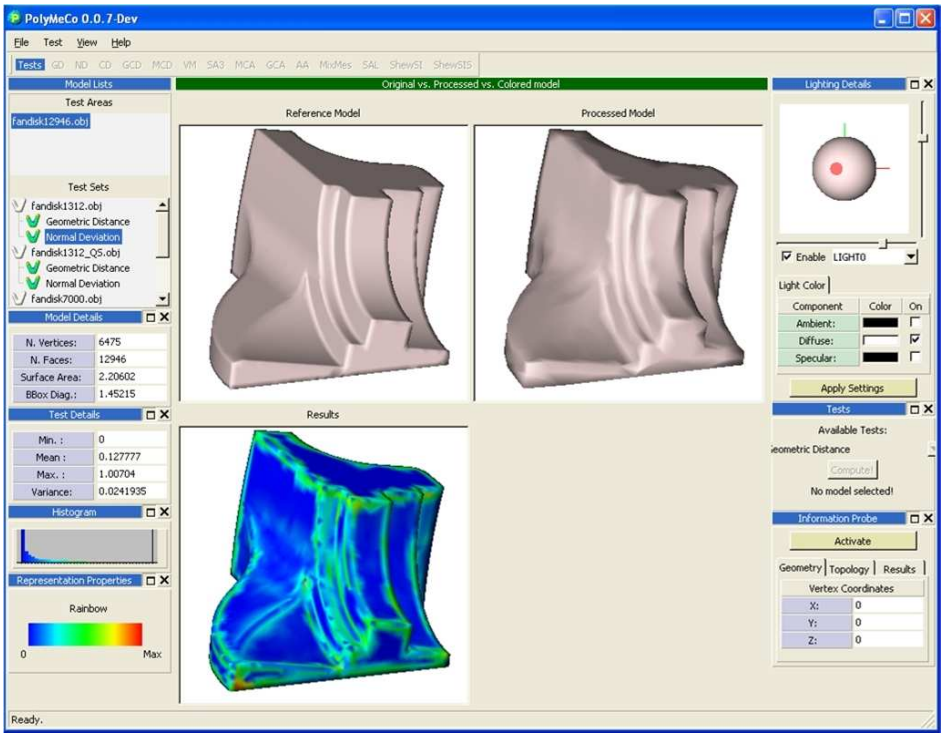



Figure 4.15: Original vs. Processed vs. Colored model *Visualization Mode*.

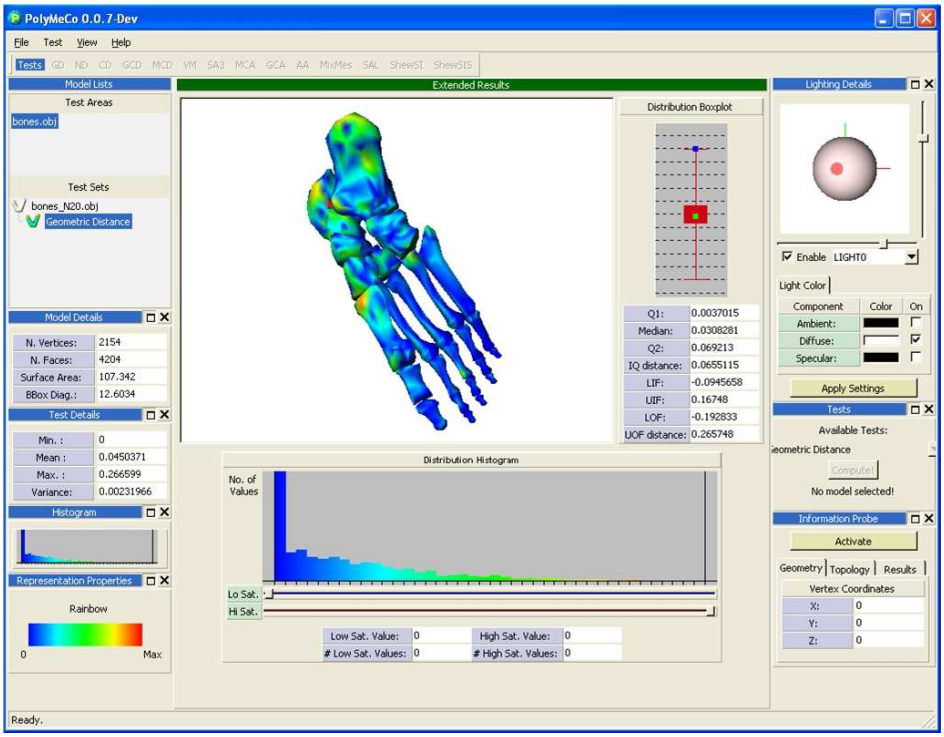



Figure 4.16: Extended results *Visualization Mode* showing a colored model, a histogram and a boxplot for data obtained with the Geometric Distance *Computational Measure*.

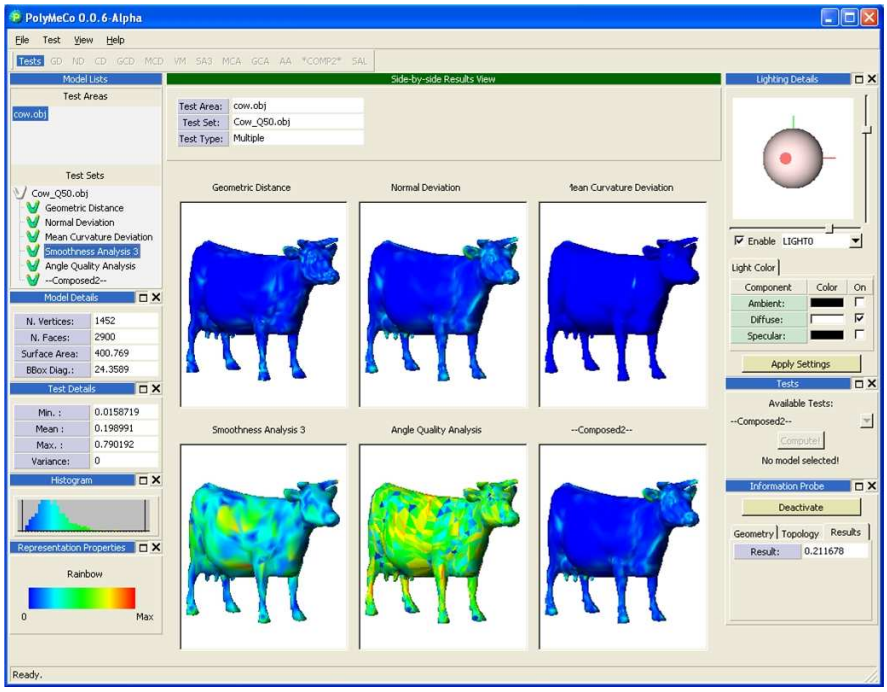



Figure 4.17: Simultaneous visualization of colored models regarding results obtained using: top, Geometric Distance (left), Normal Deviation (center), and Mean Curvature (right); bottom, Smoothness Analysis, Triangle Quality and Mixed Measure.

### 4.7.3 Simultaneous Visualization of Different Measures

This *Visualization Mode* allows the user to visualize the data distributions obtained with different *Computational Measures*, regarding one model. This can help the user to have a clearer idea of the obtained results, and provides the possibility of comparing between them and finding, for example, similarities in their behavior. The presented models remain synchronized, i.e., with similar position, orientation and size. Figure 4.17 illustrates this *Visualization Mode*.

### 4.7.4 Feature Comparison

This *Visualization Mode* allows the visualization of data distributions obtained with the same *Computational Measure* for several processed models. This can get handy in situations where we want to study different processing algorithms and compare the obtained results. Feature comparison is possible using colored models (Fig. 4.18), histograms (Fig. 4.19) and boxplots (Fig. 4.20).

PolyMeCo only allows feature comparison of models belonging to the same *Test Area* and it is up to the user to judge for which models this feature comparison can be used. For example, it may make no sense, when comparing models simplified using different simplification methods, to compare between models at different simplification levels. The models to compare can be chosen by clicking on the desired *Computational Measure* and then selecting

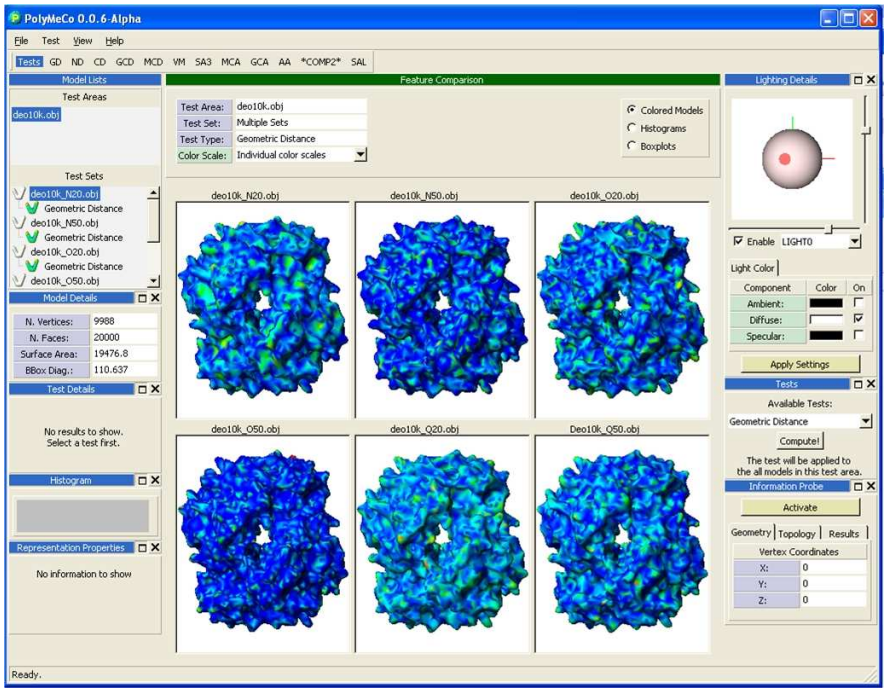



Figure 4.18: Feature Comparison *Visualization Mode*: simultaneous visualization of colored models regarding results obtained using the Geometric Distance for six simplified versions of a model.

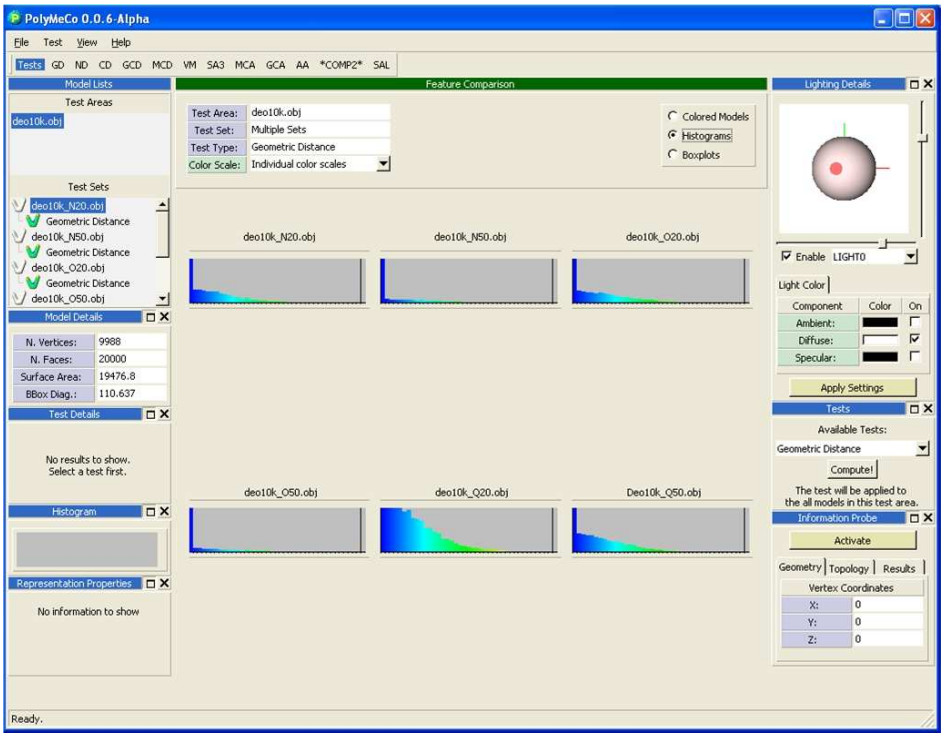



Figure 4.19: Feature Comparison *Visualization Mode*: simultaneous visualization of histograms regarding results obtained using the Geometric Distance for six simplified versions of a model.

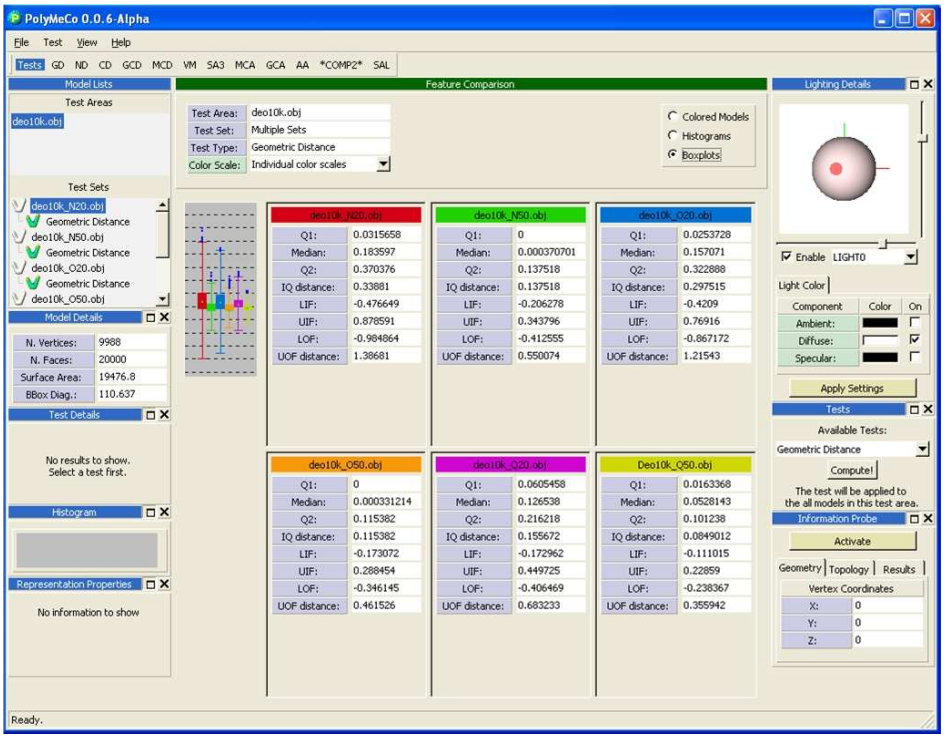



Figure 4.20: Feature Comparison *Visualization Mode*: simultaneous visualization of boxplots regarding results obtained using the Geometric Distance for six simplified versions of a model.

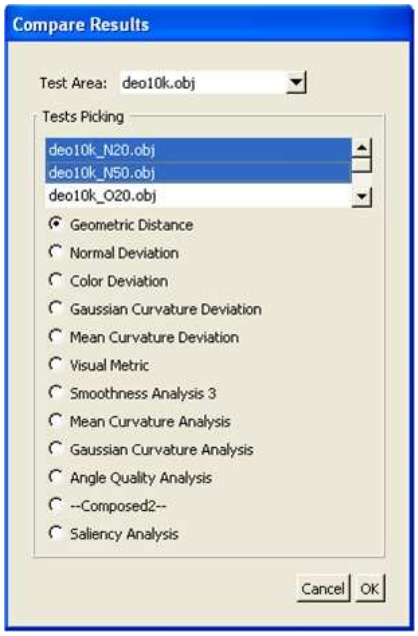
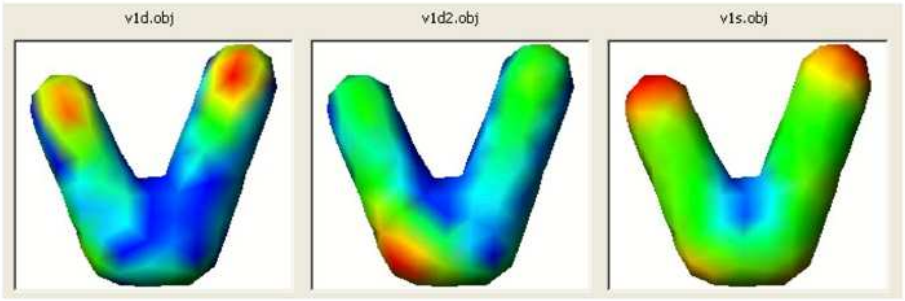
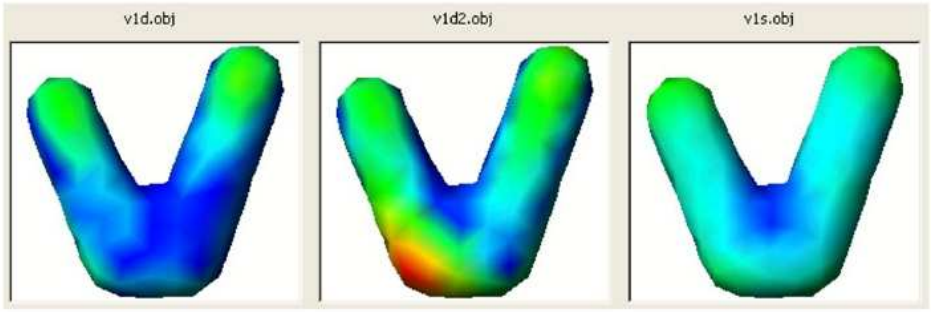



Figure 4.21: Dialog box which allows selecting models to add to the Feature Comparison *Visualization Mode*.

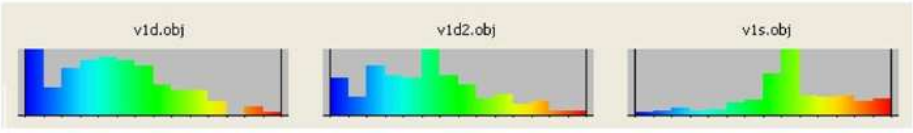



(a)

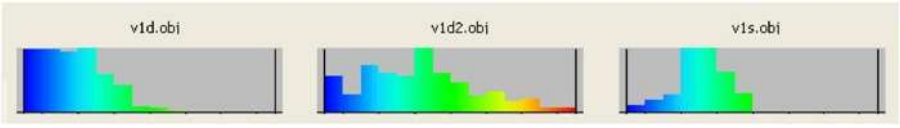



(b)

Figure 4.22: For different models, comparison of the results obtained by computing the same measure using colored models: (a) each model was colored using an individual color map; (b) all models were colored using a common color map allowing direct comparison among them.




(a)




(b)

Figure 4.23: Comparison using histograms of the data obtained by computing the same measure for different models: (a) the histograms are drawn using individual ranges; (b) all histograms are drawn using a common value range allowing direct comparison among them.

from the set of models for which it was computed in the current *Test Area* (see Fig. 4.21).

When using this *Visualization Mode* one must not forget important details. Each one of the colored models (and histograms) uses the full range of the color scale to represent the data, i.e., for each colored model the last color of the color scale is used to represent the maximum value obtained for that particular model. This kind of coloring can still be useful to compare value incidence in each model, but it is not possible to directly compare distributions using the colored models.

In order to allow this kind of comparison (not possible with the other comparison tools described earlier), PolyMeCo provides the option of using a common color map on all the models. The last color of the color scale is now used to represent the maximum value obtained among all the compared models. The same happens with the histograms, i.e., they are redrawn considering the new maximum.

Figure 4.22a allows comparing results using an individual color map for each model. Notice that it is not clear which is the model that has higher associated values. On Fig. 4.22b a common color map is used thus allowing for an easier and more accurate comparison.

The same happens with histograms: again, direct comparison among them can only be correctly performed when using a common value range. Figure 4.23 shows the histograms drawn using their individual ranges, and then using a common value range.

## 4.8 Information Windows

It is also important to provide users with additional information about what they are visualizing. This is done using *Information Windows*.

An *Information Window* (see Fig. 4.24) takes information about the currently active models and about scene properties and provides information regarding the current viewed items. It can also provide graphical user interfaces for the configuration of scene parameters

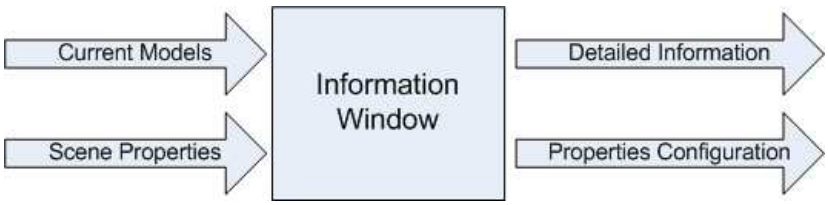



Figure 4.24: Diagram showing inputs and outputs for an *Information Window*.

(e.g., light source properties) or auxiliary tools to obtain information useful in the current context, like an interrogation tool when visualizing a particular *Representation*.

An *Information Window* can also provide a shortcut to particular operations that can be accomplished in the current context. For example, it can allow a user to select and compute a particular measure for the currently selected model. These *Information Windows* can be moved by the user, docked inside PolyMeCo's window, or hidden when not being used. Undocking or hiding *Information Windows* can be useful to maximize the screen area used for model rendering, thus providing a better visualization of the presented models. Figure 4.25

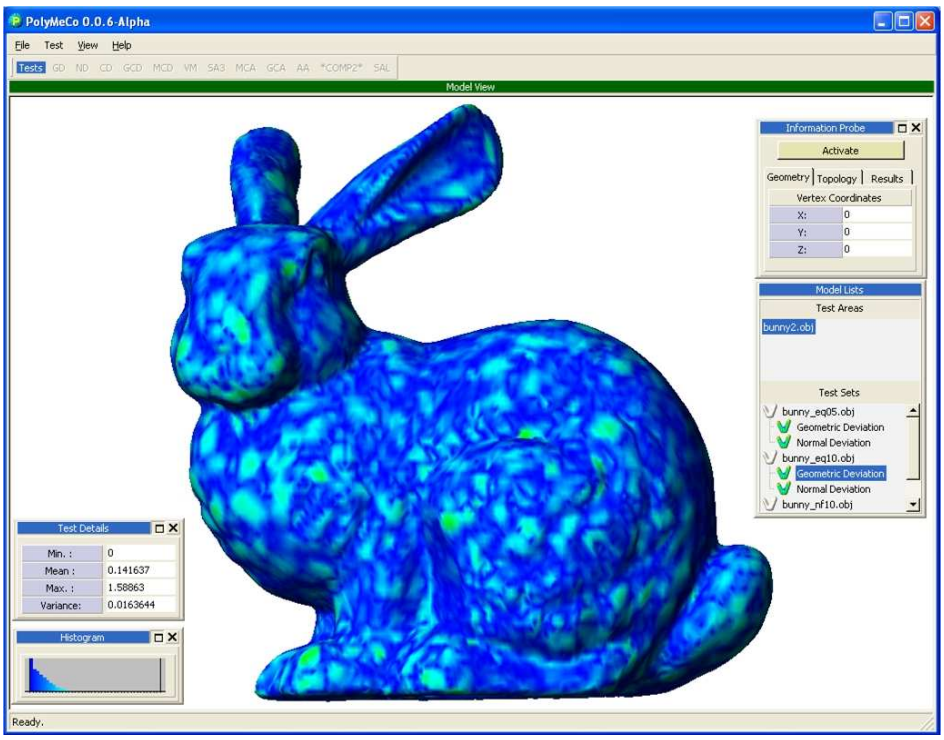



Figure 4.25: A view of PolyMeCo's user interface. All *Information Windows* have been undocked (and repositioned) and some were hidden in order to maximize the screen area used for model rendering.

shows the same colored model presented in Fig. 4.1, but now all *Information Windows* have been undocked and some, not being used, were hidden. Notice how the rendering area has increased significantly.

In what follows, a brief description of each *Information Window* provided by PolyMeCo is performed.

### 4.8.1 Model List

This *Information Window* (see Fig. 4.26 allows the user to navigate across all the loaded models and computed data. At the top it provides a list of the created *Test Areas*, i.e., the original models that have been loaded to be used as references during mesh comparisons. At the bottom, a list of all the *Test Sets* belonging to the currently selected *Test Area* is presented, i.e., all the processed versions loaded to be analysed/compared. Under each *Test Set* a list of all *Computational Measures* already computed is provided. This list can be collapsed by double-clicking on the *Test Set* name, thus allowing to hide it when it is not being necessary, leaving more space to view other items of the *Test Set* list.

This is the only *Information Window* that cannot be hidden.

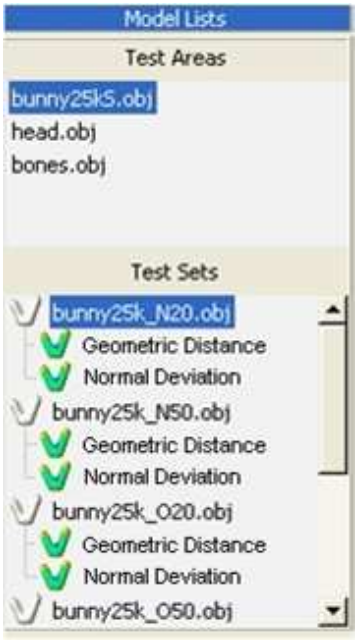



Figure 4.26: Model List *Information Window*. A list of the models opened for analysis and comparison is shown.

### 4.8.2 Model Details

This *Information Window* (see Fig. 4.27) provides some basic information about the current mesh features, namely: number of vertices, number of faces, surface area and bounding box diagonal. These values allow the user to perform a simple comparison among models according


Figure 4.27: Model Details *Information Window*. Some details about the currently selected model are presented.

to their complexity (number of vertices and faces), their size (bounding box diagonal), and surface area.

### 4.8.3 Test List

This *Information Window* (see Fig. 4.28) provides a list of all the available *Computational Measures*. The user can then select one of them and compute it for a particular model or for all models included in a *Test Area*. A short message in the bottom of the *Information Window* informs the user about which models will be included in the computation (according to the model currently selected): if the *Test Area* is selected, the *Computational Measure* will

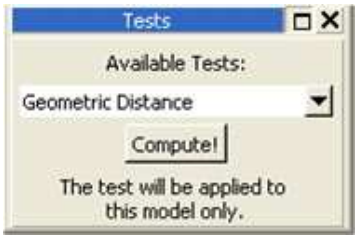



Figure 4.28: Test List *Information Window*. A list of the available *Computational Measures* is presented and the user can select and compute them for the opened models.

be computed for all associated *Test Sets*; if a *Test Set* is selected then the *Computational Measure* will only be computed for that particular model. This is an important feature to allow systematic use of PolyMeCo as a mesh evaluation tool as it can speed-up the task of computing several measures for several models, as opposed to what is possible with other tools described in the literature, which only allow working with a model and *Computational Measure* at a time.

### 4.8.4 Test Details

In this *Information Window* (see Fig. 4.29) some numerical values appear which summarize the results obtained with a particular *Computational Measure*, namely: minimum, mean, maximum and variance values.

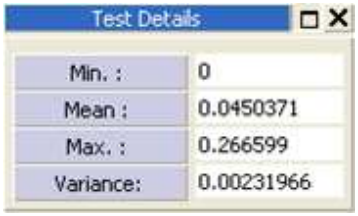



Figure 4.29: Test Details *Information Window*. Some numerical details about the results currently being analysed are provided.

### 4.8.5 Small Histogram

This *Information Window* (see Fig. 4.30) shows a small histogram, colored according to the currently used representation, which allows the user to obtain a rough idea of how the values obtained with a computational measure are distributed. A larger histogram can be viewed in the Extended Results *Visualization Mode* described in Section 4.7.2.

### 4.8.6 Representation Properties

This *Information Window* provides information regarding the currently used *Representation*. When visualizing a model colored according to the data obtained using a *Computational Measure* this *Information Window* shows the used color scale (Fig. 4.31, left). When visualizing

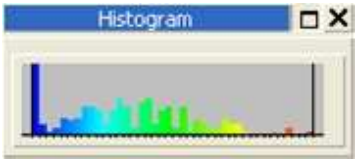



Figure 4.30: Small Histogram *Information Window*. A small histogram which summarizes the data set currently being analysed is presented.

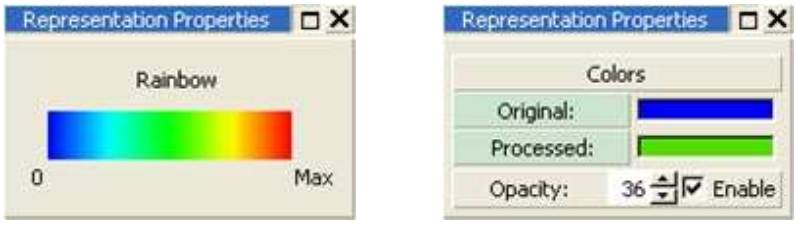



Figure 4.31: Representation Properties *Information Window*. Left, the color scale used in a *Representation* is provided; right, the properties for the Model Superposition *Representation* are presented.

other representation it may provide several controls which allow modifying some of its properties. The Model Superposition *Representation* is such a case. The user can change the color used to render each model, set an opacity level for the original model and enable/disable transparency (Fig. 4.31, right).

### 4.8.7 Location Probe

This *Information Window* (see Fig. 4.32) provides a probe tool which allows the user to obtain information regarding a particular mesh vertex. The provided information concerns geometry (vertex coordinates), topology (vertex neighborhoods and their properties) and associated data values, such as those obtained using a particular *Computational Measure*.

Figure 4.32 shows some details about the Location Probe Information Window and its usage. When a mesh vertex is selected, a green sphere appears at its location along with an highlight of its neighbors (Fig. 4.32, top). The user can define the number of visible neighborhoods on the Topology tab (Fig. 4.32, bottom center) and read information about each neighborhood (one-ring, two-ring, etc.) such as the number of vertices it includes. The information about vertex coordinates can be found in the Geometry tab (Fig. 4.32, bottom left) and the value computed for it, using one of the *Computational Measure*, is presented in the Results tab (Fig. 4.32, bottom right).

The Location Probe still does not provide obtaining information about mesh faces (e.g., area, Triangle Quality value).

### 4.8.8 Lighting Details

This *Information Window* (see Fig. 4.33) provides the user with several features to view/change the light properties of the scene. It is possible to increase the number of light

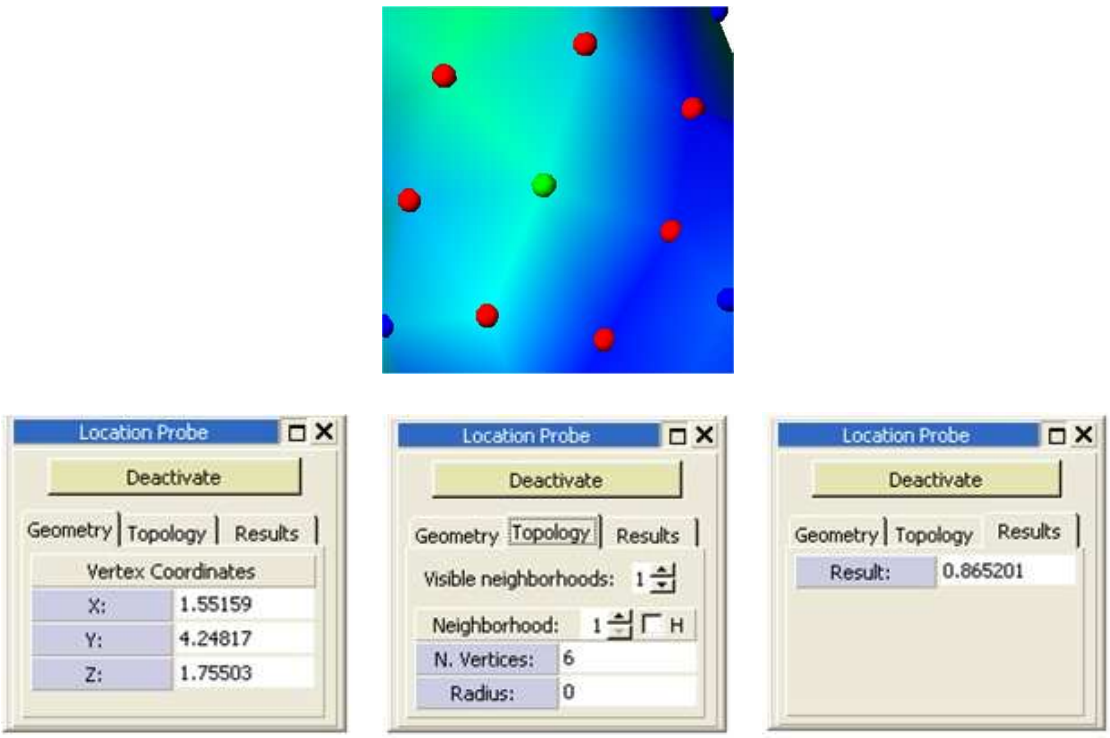



Figure 4.32: A vertex was selected with the Location Probe. Top, mesh surface detail showing the selected vertex along with its direct-neighborhood. Bottom, the three tabs on the Location Probe *Information Window* showing information about geometry, topology and results associated with the selected vertex.

sources up to three and control their position as well as their ambient, diffuse and specular components. Light source orientation is always towards the origin, as can be seen in the representation appearing on the top of the *Information Window* where cones represent the light sources and a sphere represents the illuminated object. The chosen light source properties are applied to the sphere thus allowing the user to have a better perception of the results. Due to performance issues, specially when working with complex models, light changes are only applied to the model being visualized when the user presses the *Apply Settings* button.

In order not to interfere with the *Representations* which use color (light source color could change perceived mesh surface colors), the light source properties for those scenes are predefined and cannot be changed. The light source properties set by the user are only applied when viewing original or processed models.

This feature has been used to better examine and compare surfaces of models obtained using different simplification methods, as described in Chapter 5.

## 4.9 Additional Features

As PolyMeCo is an integrated environment, allowing the simultaneous presence of several models and computed mesh properties, it is desirable to provide features which help the user to deal with the associated complexity and the obtained data.

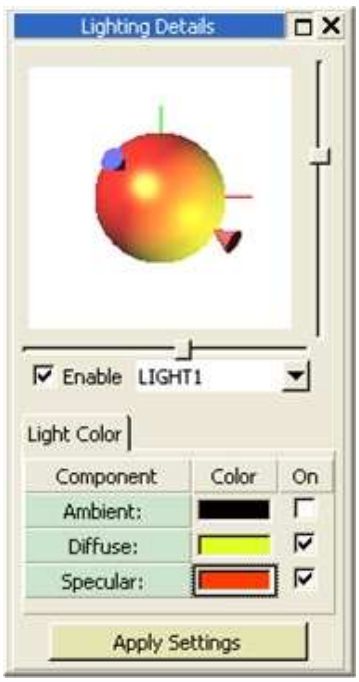



Figure 4.33: Lighting Details *Information Window* where users can customize light source properties.

Regarding this issue, several features, which are described next, are available in PolyMeCo.

### Enable / Disable rendering

When working with complex models, the required rendering times may be large. For this reason, PolyMeCo provides an option to disable model rendering. In this way, navigating along the models to compute intrinsic properties or deviation measures is faster. At any time the user can activate model rendering to analyse the results using one of the available *Representations*.

### 3D Representation Snapshot

Sometimes, it can be important to capture a particularity seen in a *Representation* in order to view it later or to illustrate a conclusion. PolyMeCo provides the possibility of taking a snapshot of a *Representation* saving it in BMP, GIF or JPEG formats.

### Workspace Saving/Loading

It is also possible to save the information regarding the current workspace, i.e., the opened models, the selected measures and the computed data, etc. This is done by creating a XML [2] file containing all this information.

The choice of the XML format was due to the fact that it allows an easy data interchange and is plataform/application independent. Is also has the advantage of being user readable.

Another particularity is that by writing a XML style sheet it is possible to view the results contained in the XML file in a structured way, using a web browser (Fig. 4.34). This allows the readability of the results even without PolyMeCo.

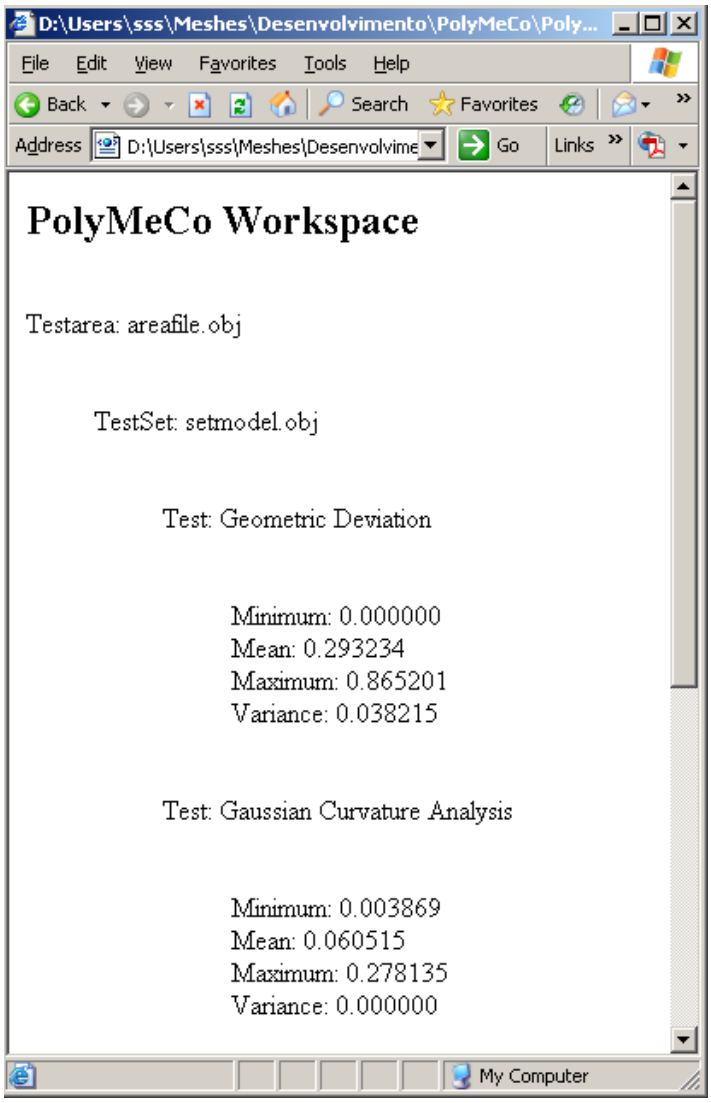



Figure 4.34: PolyMeCo's workspace information viewed on a web-browser.

## 4.10 Conclusion

In this chapter PolyMeCo, a tool for mesh analysis and comparison, has been presented. After describing a possible pipeline for mesh analysis and comparison, several PolyMeCo features were presented, namely, the supported *Computational Measures*, *Representations*, *Visualization Modes* and *Information Windows*.

PolyMeCo provides several features which were missing in other mesh analysis and comparison tools described in the literature. These features allow the user to gain greater insight into the data obtained using several *Computational Measures*, while having an integrated en-

vironment where he/she can examine model surfaces (with the chance of customizing light source properties), navigate across loaded models and restore previous workspaces. By being allowed to freely navigate between *Visualization Modes* and change the used *Representations*, the user can experiment with the different features without loosing focus, thus contributing for an easier exploration of all the provided information.

As always, the applicability of a software tool depends on the features it provides but also on how these features can be used to attain particular goals. For that purpose, two application examples will be presented on the following chapter.

# Chapter 5

# Application Examples

In many polygonal mesh applications the original meshes (e.g., obtained using 3D scanners) are very complex and do not allow proper visualization and interaction (e.g., in low resources systems like *PDAs*), or their transmission through a network. In these cases the solution can be mesh simplification.

Many simplification algorithms have been proposed in the literature (see Chapter 3) but only a few studies compare those methods with each other, helping users to choose the proper method for a specific application (examples of such studies are those of Cignoni et al. [36] and Rogowitz et al. [117]). For the purpose of mesh comparison, some metrics and tools have been proposed in the literature. How these metrics estimate user perceived quality is still an unanswered question, but some authors as Watson et al. [159] and, more recently, Silva et al. [138, 134] and Sousa Santos et al. [143], who compare simplification methods using observer studies, aim to produce guidelines which help users understand not only the physical implications of a particular metric, but also how it can be used to estimate perceived quality.

PolyMeCo can play an important role in that kind of tasks due to its convenient environment offering access to mesh analysis and comparison methods, as well as making available several visualization tools.

This chapter presents two application examples of PolyMeCo. In the first example, PolyMeCo was used to compute several intrinsic properties and difference measures of a set of polygonal models. The obtained data was then compared with quality results gained from observer studies in order to explore if any of the measures available in PolyMeCo could be used to estimate visual mesh quality as perceived by users. Although this application does not explore the visualization capabilities provided by PolyMeCo, as it only uses the mean values of the computed measures, PolyMeCo allowed computing the same computational measure for several models at once, which is much easier than using any other tool available.

In the second example, PolyMeCo was used to compare two mesh simplification algorithms, *QSlim* and *NSA*. The visualization features of PolyMeCo were explored aiming, in a first instance, for a faster and more qualitative evaluation followed by a careful (quantitative) analysis of the outputs.

## 5.1 Case Study 1 – Quality Evaluation of Polygonal Meshes Using an Observer Study and Quality Indices

Following on earlier work carried out by the author and his colleagues, described in Sousa Santos et al. [143], but only for mesh models of the lungs, a comparison among three simplification methods is presented using (1) an observer study involving 65 observers and (2) quality indices computed using PolyMeCo.

The main goal was to ascertain whether the findings previously obtained for lung models, through quality indices and a study with 32 observers, could be generalized to other types of models and confirmed for a larger number of observers. In particular, it should be verified whether the Geometric Distance was a good estimator of user perceived quality for severe simplifications while the Normal Deviation was a good quality estimator for moderate simplifications.

In what follows the observer study and obtained results will be presented. Next, these results will be compared with those obtained using quality indices computed using PolyMeCo.[1]

### 5.1.1 Observer Study

The observer study — whose main features, as well as the experiment, are presented in what follows — was set up and carried out exactly as it had been done before for the study using lung models: see [138] for a thorough description of the objectives, context, framework, experimental methodology and data analysis of that former study.

Note that the former observer study was a suitable testbed to confirm that the developed experimental design and protocol allowed perceived quality evaluation, as well as to establish the methods for the statistical analysis of the collected data.

#### Main features

We intended to compare three mesh simplification methods — the widely used *QSlim* [48, 49] and two other methods provided by the *OpenMesh* [28] library (one using error quadrics, the other additionally using a normal flipping criterion) — regarding the *"perceived quality"* of the resulting meshes, for a set of five reference models of different kinds (see Fig. 5.1 and Table 5.1), and for two simplification levels: severe (to 20% of the original number of mesh faces) and moderate (to 50%).

Test sets were built from the set of five reference models: for each model and for each simplification level (severe and moderate) three simplified models were created using the three simplification methods. This resulted in a total of 10 test sets, each composed by the original and the three simplified models (five sets for each simplification level). Note that the five models chosen are different from each other and have different numbers of vertices and faces.

[1] The work described in this section has been published in Silva et al. [137].

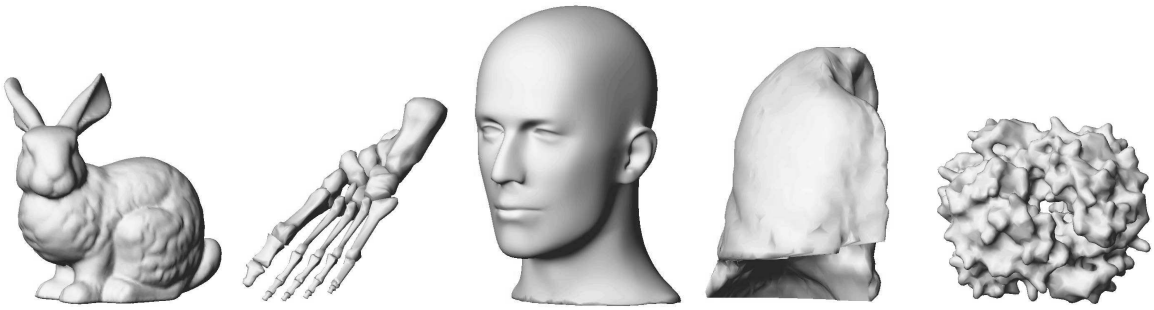


Figure 5.1: Original models used. From left to right: Bunny, Foot, Head, Lung and Strange.

| **Models** | **# Vertices** | **# Faces** |
|---|---|---|
| Bunny | 25,103 | 49,999 |
| Foot | 2,154 | 4,204 |
| Head | 11,703 | 23,402 |
| Lung | 4,811 | 9,514 |
| Strange | 9,988 | 20,000 |

Table 5.1: Some details regarding the models used in the experiment.

Starting from the hypothesis that distinct mesh simplification methods have different effects on the model quality perceived by human observers, possibly varying with the simplification level and other factors, we assessed that by asking for the observers' preferences and ratings, which are widely used to obtain relative judgements from observers and are probably the most adequate indices of fidelity [159].

With preferences, each observer assigned to the three simplified models in a test set an ordering according to their perceived quality, regarding the original reference model. With ratings, each observer classified each simplified model regarding the reference model (using a rate from 1 to 5), according to its perceived quality. For each of these tasks, the time taken to reach a decision and the number of interactions (performed on each model before deciding) were also recorded, since they seemed to be related to the degree of difficulty observers encounter in performing the preference and rating tasks.

To allow an easy implementation of the experimental protocol, as well as an easy storage and management of the collected data, the same software application that had been developed for the former lung models study was used [138]. Note that, with this application, observers were free to interact with a model, by changing its position, orientation and scaling factor, and choose the viewpoints they wished to analyse a model from, which is a more realistic and less limitative setting than the one used by Watson et al. in a similar study [159].

### The experiment

A within subjects experimental design was used [40], i.e., each observer performed under each different condition. Due to the possible influence of learning effects, nervous behavior in the first task or fatigue in the last, all test sets were presented randomly to each observer and, for

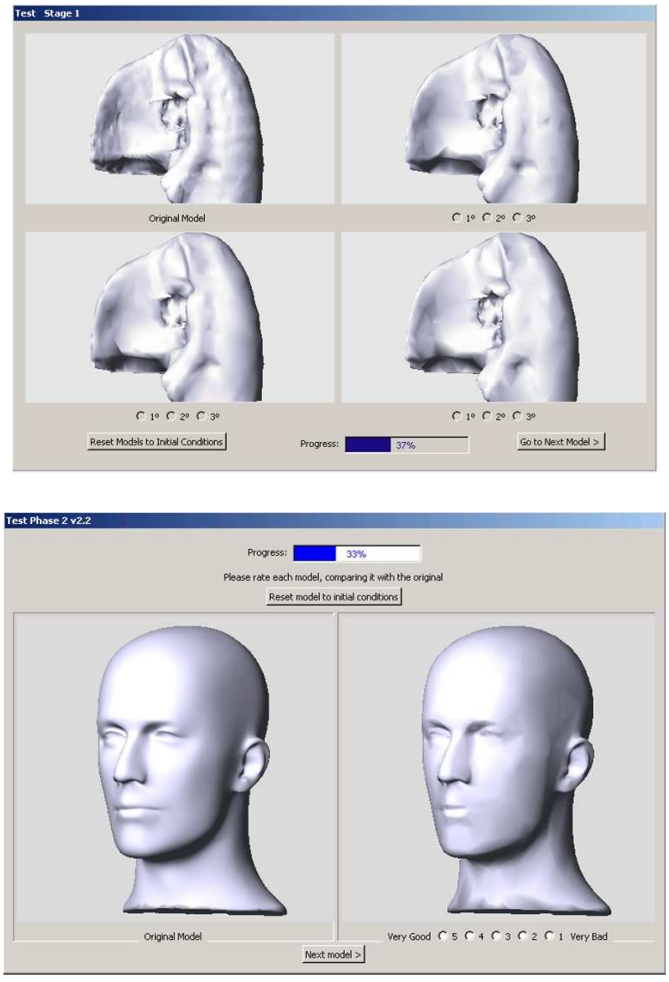



Figure 5.2: First phase (top): the original model (upper left) and three simplified versions are presented and preferences are asked. Second phase (bottom): the original model (left) and one simplified version are presented and a rating is asked.

each observer, the order of presentation of the models, within each set, was randomly chosen.

For each observer, the experiment was divided into two phases (see Fig. 5.2):

In the first phase — preference task —, an observer was sequentially presented with each one of the 10 test sets and asked to assign a first, second and third place to each of the simplified models, according to their perceived quality regarding the original.

In the second phase — rating task —, an observer was sequentially presented with an original model and one of its simplified versions, taken from one of the test sets, and asked to rate the simplified model using a five level Likert scale [16] from 1 (very bad) to 5 (very good), once again based on its perceived quality.

Sixty-five engineering students and lecturers, aged between 18 and 55 (the majority, 45 individuals, was between 18 and 25 years old), participated in the experiment (57 men and 8 women). Forty-one individuals declared to have experience in viewing/manipulating 3D models. Table 5.2 lists, for each observer, the collected data for each experiment phase. Since

| First Phase | Second Phase |
|---|---|
| Preferences (first, second, third) | Ratings given to each model |
| Order of presentation of the test data sets | Order of presentation of the simplified versions of the models |
| Number of interactions with each test data set | Number of interactions with each simplified model |
| Time taken to order each data set | Time taken to rate each simplified model |

Table 5.2: Type of data collected on both experiment phases.

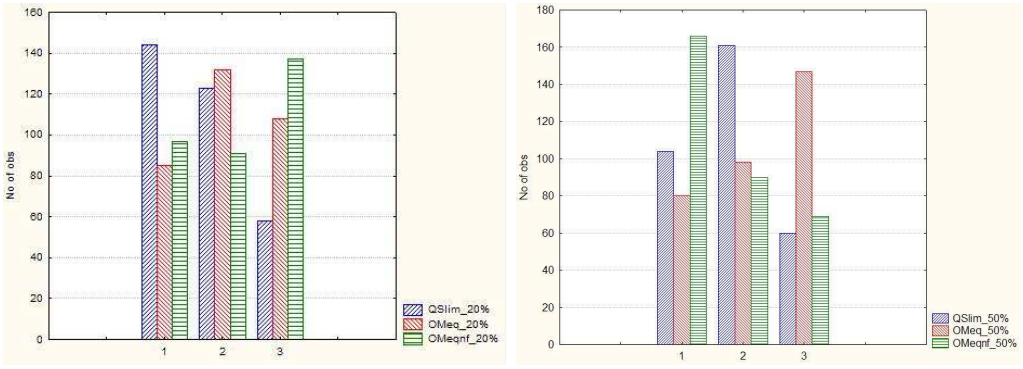



Figure 5.3: Preferences corresponding to the two simplification levels: severe (left) and moderate (right).

the gender, age or experience with 3D object manipulation of an observer might influence the results, this information was used to characterize the profile of each observer.

### Results – Phase 1

Concerning the preferences, as a first step, bar charts where built showing the number of first, second and third places obtained by each simplification method for the two simplification levels, as shown in Fig. 5.3 (severe simplification on the left and moderate simplification on the right). The bar-chart on the left seems to reveal a tendency of the observers to prefer the simplified versions using *QSlim* (larger number of first places), then the versions simplified using *OpenMesh* and, in third place, the versions simplified using *OpenMesh* with normal flipping. The bar-chart on the right of Fig. 5.3 seems to reveal that the observers prefer the simplified versions using *OpenMesh* with normal flipping, followed by *QSlim* and *OpenMesh*.

In order to confirm the statistical significance of the above-mentioned tendency, contingency tables [70] were used (see Table 5.3) and the independency of hypothesis were tested. The independency between the simplification method and the observers' preference was rejected for both simplification levels, with $\chi^2 = 57,57 >> 9,49$ ($\chi^2$(4d.f.;$\alpha = 0.05$)) for severe simplifications and $\chi^2 = 110,54 >> 9,49$ ( $\chi^2$(4d.f.;$\alpha$=0.05)) for moderate simplifications. These results suggest that observers are indeed responsive to the simplification method used, although they react in a different way according to the simplification level; for severe simplifications *QSlim* obtains the best results, while for moderate simplifications it is *OpenMesh* with normal flipping that obtains most of the first places.

The results obtained by the contingency tables can be visualized using a Correspondence Analysis [70]. Figure 5.4 shows the factorial planes corresponding to the contingency tables for

Severe Simplifications (20%)

| | $1^{st}$ | $2^{nd}$ | $3^{rd}$ |
|---|---|---|---|
| **QSlim** | 144 | 123 | 58 |
| **OpenMesh** | 85 | 132 | 108 |
| **OpenMeshNF** | 97 | 91 | 137 |

Moderate Simplifications (50%)

| | $1^{st}$ | $2^{nd}$ | $3^{rd}$ |
|---|---|---|---|
| **QSlim** | 103 | 162 | 60 |
| **OpenMesh** | 80 | 98 | 147 |
| **OpenMeshNF** | 166 | 90 | 69 |

Table 5.3: Contingency table corresponding to preferences for both simplification levels.

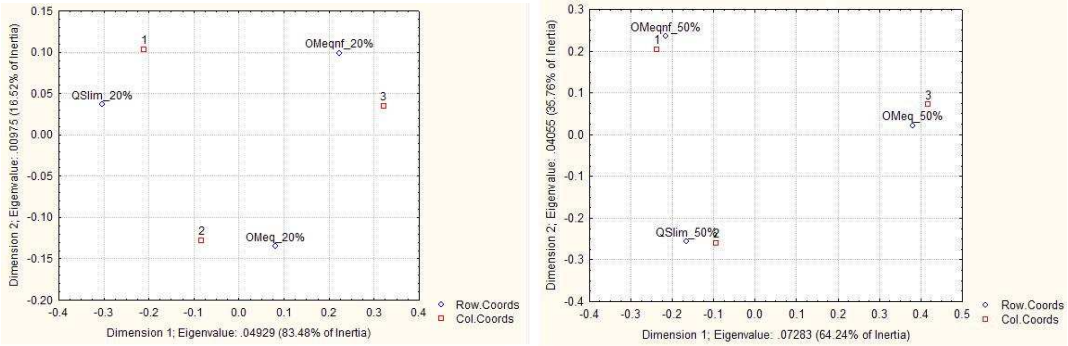



Figure 5.4: Correspondence Analysis (where simplification methods are in rows and preferences in columns) for the two simplification levels: severe (left) and moderate (right).

both simplification levels. In these projections we can observe that, for severe simplifications each simplification method is clearly associated with a type of preference: first place for *QSlim*, second place for *OpenMesh* and third place for *OpenMesh* with normal flipping. A different and even stronger association appears for moderate simplifications: *OpenMesh* with normal flipping is associated with the first place, *QSlim* with the second and *OpenMesh* with the third place. A thorough description and analysis of the results obtained in the observer study can be found in Silva et al. [134].

### Results – Phase 2

Concerning ratings, as a first step, we produced bar charts showing the number of marks (1 – very bad, to 5 – very good) obtained by each simplification method for the two simplification levels as shown in Fig. 5.5 (severe simplification on the left and moderate simplification on the right). The bar-chart on the left seems to reveal a tendency of the observers to rate poorly all the simplification methods, specially the simplified versions using *OpenMesh* with normal flipping (larger number of ones), then the versions simplified using *OpenMesh* and *QSlim*. It must be noted that almost nobody rated above 4, and even for this mark the number of observations is very low. On the other hand, there is a visible increase on the rating, when the level of simplification decreases (moderate simplification level). The bar-chart on the right shows a majority of marks ranging from 3 to 5. All the three methods seem equally well rated, perhaps with a slight advantage (larger number of fives) of the *OpenMesh* with normal flipping, which was considered the worst on the severe simplification level. This result is consistent with the one previously obtained from the preferences.

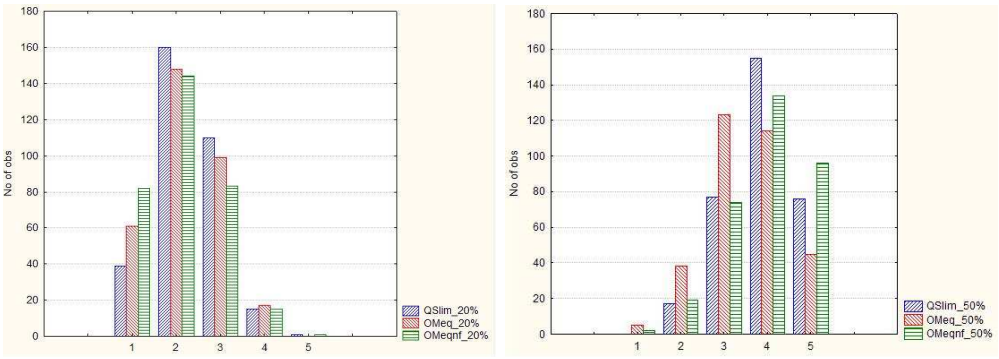



Figure 5.5: Ratings corresponding to level of simplification: severe (left) and moderate (right).

Severe Simplifications (20%)

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **QSlim** | 36 | 154 | 106 | 16 | 3 |
| **OpenMesh** | 61 | 148 | 99 | 17 | 0 |
| **OpenMeshNF** | 82 | 144 | 83 | 15 | 1 |

Moderate Simplifications (50%)

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **QSlim** | 0 | 17 | 77 | 155 | 76 |
| **OpenMesh** | 5 | 38 | 123 | 114 | 45 |
| **OpenMeshNF** | 2 | 19 | 74 | 134 | 96 |

Table 5.4: Contingency table corresponding to the ratings for the two simplification levels: severe (20%) and moderate (50%).

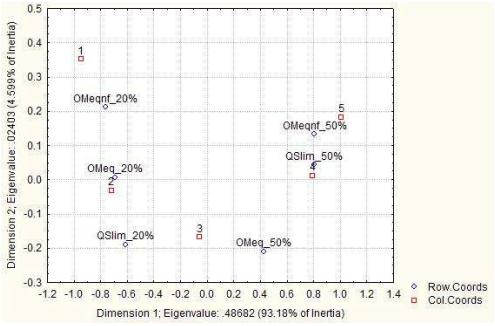



Figure 5.6: Correspondence Analysis (where simplification methods are in rows and ratings are in columns) for the two simplification levels: severe and moderate.

As in the first phase, in order to confirm the statistical significance of the above-mentioned tendency, contingency tables were used (Table 5.4) and the independency of hypothesis were tested. The independency between the simplification method and the observers' ratings was rejected for the severe simplifications, with $\chi^2$=24,57>15,5 ( $\chi^2$(8d.f.;$\alpha$=0.05), as well as for moderate simplifications, $\chi^2 = 1013,54 >> 31,41$ ($\chi^2$(20d.f.;$\alpha$=0.05). These results suggest that, for this task as for the preferences task, observers are responsive to both simplification method and simplification level.

The visualization of the contingency tables using a Correspondence Analysis (Fig. 5.6) shows that for the moderate simplifications all methods obtain similar ratings and higher than the obtained for the severe simplifications.

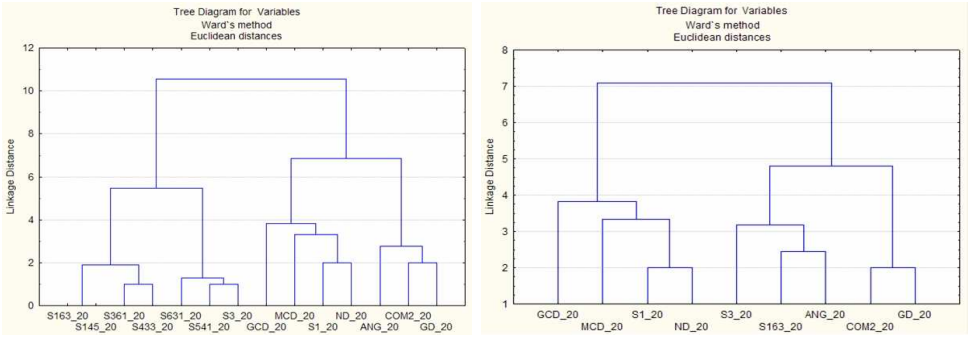



Figure 5.7: Left, dendogram showing associations between all computed quality indices for **severe** simplifications; right, dendogram showing associations after discarding some redundant quality indices.

### 5.1.2 Quality Indices

Several quality indices provided by PolyMeCo were computed for each one of the 30 simplified models (5 models $\times$ 3 simplification methods $\times$ 2 simplification levels), namely: Geometric Distance (GD), Normal Deviation (ND), Mixed Measure (COM2), Mean and Gaussian Curvature Deviations (MCD and GCD), Angle Analysis (ANG) and Smoothness. For this particular purpose some variations where considered in the Smoothness measure. S3 is the usual smoothness measure which computes the distance of each vertex to the centroid of its one ring. S1 computes smoothness using, not the norm of the vertex-centroid distance, but the difference vector, which implies that, when computing the mean value, symmetric differences will cancel each other. This tries to explore the fact that, if a model has a less smooth surface, it can still be deemed acceptable if it maintains some symmetry. Finally, the $\mathbf{S}ijk$ indices compute smoothness using vertices from the first, second and third neighborhoods of a vertex, assigning them weights of $0.1 \times i$, $0.1 \times j$ and $0.1 \times k$, respectively.

#### Results

For every simplified model, the mean value for each one of the above described quality indices was computed. Based upon those mean values, for each simplified model and for the same simplification level, a rank (first, second and third place) was assigned to the three models comprising each test set.

To study the association between quality indices, and to assert possible redundancies in the results provided by different indices, Cluster Analysis [70] was used. For severely simplified (20%) models, and looking at the dendogram on the left of Fig. 5.7 (using Ward's method as a measure of proximity), it is possible to detect some associations between indices, namely across the $\boldsymbol{Sijk}$ group. A closer look at the data revealed some degree of redundancy. After discarding the indices considered redundant, a new dendogram was computed (Fig. 5.7 on the right), which is now easier to read. Notice that, as it was expected, the COM2 index appears associated with the GD for the severe simplifications. Notice also that S1 and S3 appear clearly separated in different branches.

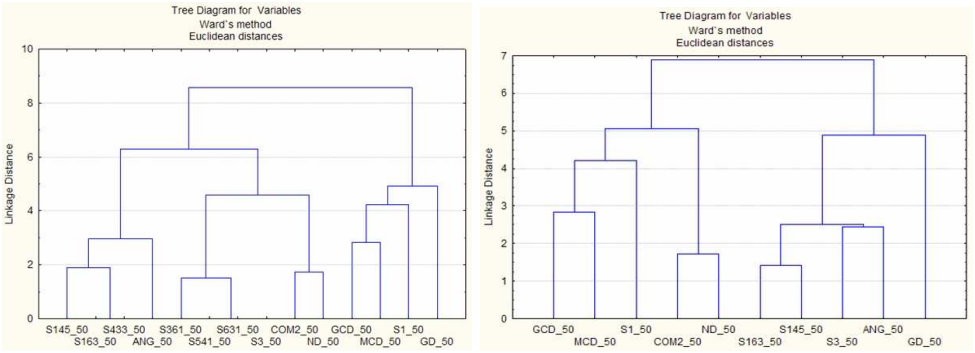



Figure 5.8: Left, dendogram showing associations between all computed quality indices for **moderate** simplifications; right, dendogram showing associations after discarding some redundant quality indices.

For the moderate simplification a similar analysis was performed. To detect possible redundancies, Cluster Analysis was used and a dendogram was drawn for all indices (Fig. 5.8, on the left). Once again some redundancy was detected and, after analyzing the obtained data, several indices were discarded and a new dendogram was drawn (Fig. 5.8, on the right). Notice that, as expected, the COM2 index now appears associated with the ND.

After this preliminary analysis, contingency tables were built for the results obtained with each quality index and for each simplification level. The results were then visualized using Correspondence Analysis [70]. Some of the resulting factorial planes are presented in Fig. 5.9, for the severely (20%) simplified models, and in Fig 5.10, for moderately (50%) simplified models.

Regarding the global performance of the quality indices, different behaviors can be identified from Figs. 5.9 and 5.10. Namely, for the GD index and for both simplification levels, the associations between method and rank are clearly presented: *QSlim* is associated with the first place, followed by *OpenMesh* in second, and *OpenMesh* with normal flipping in the third place, which indicates that *QSlim* is the simplification method producing, in general, simplified models with lower geometric distance to the original reference models, followed by *OpenMesh* and *OpenMesh* with normal flipping. For severe (20%) simplifications, ND does not present such a clear association, although that happens for moderate simplifications; a similar situation occurs with S1, but for the moderate and severe simplification levels, respectively.

Note that, for severe simplifications, for indices such as GCD and ANG there is also a clear method and rank association, but different from the association found for GD and COM2. The latter two quality indices seem to have a similar behavior, and are able to clearly discriminate between the three simplification methods (see Fig. 5.9); as said before, this association is also clearly shown in the right dendogram of Fig. 5.7.

A similar analysis can be done for the moderate simplification level, using the factorial planes depicted in Fig. 5.10. It is possible to verify, for example, clear associations between method and rank for the ND and COM2 indices, with *OpenMesh* with normal flipping associated with the first place, *QSlim* with second and *OpenMesh* with the third place. For

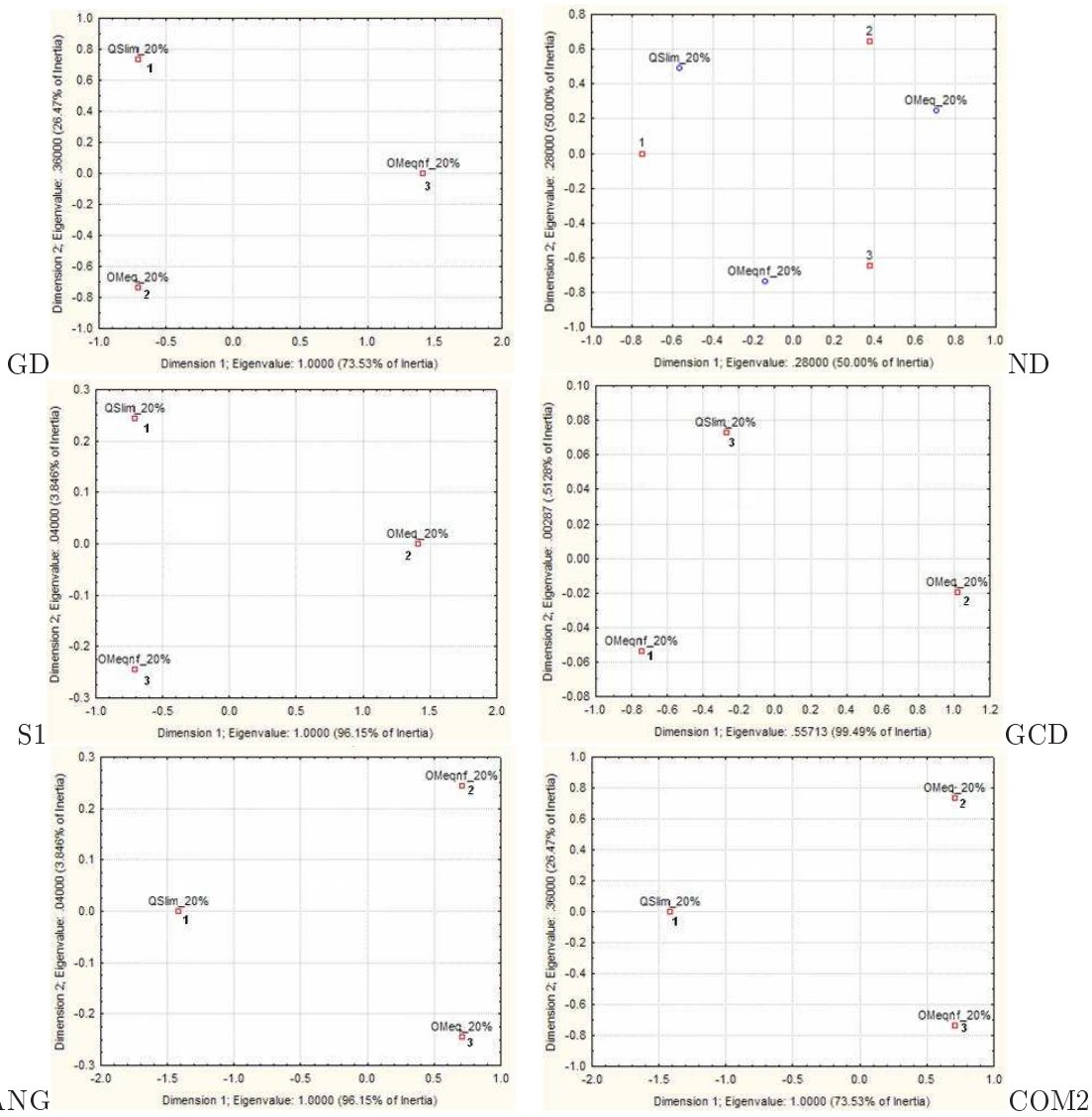



Figure 5.9: Quality indices: correspondence analysis showing the associations between simplification method and ranks obtained for **severe** simplification.

indices such as GD and GCD there are clear but different method and rank associations. Note that the ND and COM2 quality indices seem to have now a similar behavior, and are able to clearly discriminate between the three simplification methods; as said before, this association is also clearly shown in the right dendogram of Fig. 5.8.

The main conclusions that can already be mentioned are the following: GD is a quality index clearly able to discriminate between the three simplification methods but for severe simplifications only, while ND plays the same role but for moderate simplifications only; this is the behavior that had already been identified for those same quality indices when analyzing lung models [143]; the Mixed Measure index COM2, which was established based on that previous study, is able to discriminate between the three simplification methods for both simplification levels, as was expected. Note that, since the behavior of the GD and ND quality indices is globally the same for the lung models of the former study, and for the generic

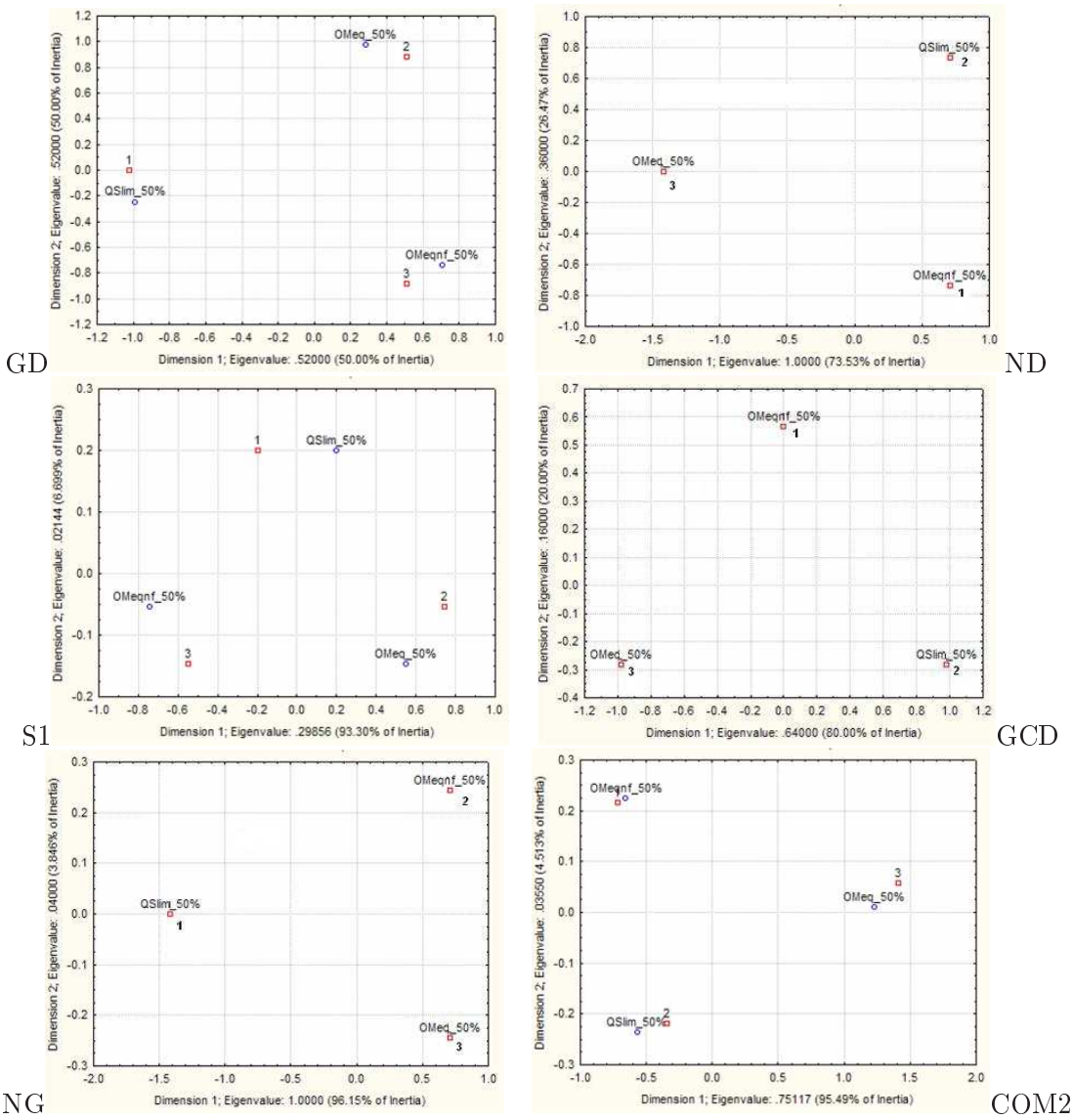



Figure 5.10: Quality indices: correspondence analysis showing the associations between simplification method and ranks obtained for **moderate** simplification.

models of the current study, it seems that simplification method and level are the determinant factors, not the type of models being simplified.

### 5.1.3 Comparison of Global Results

As the used quality indices provide information about differences between two meshes (a clue towards the quality of the simplified models), we compared the results thus obtained with those provided by the observer study, and tried to identify the quality index which better estimates model quality, as it is perceived by users.

By comparing the factorial plane on the left of Fig. 5.4, showing the associations between simplification method and observer preferences for the severe simplifications, with the factorial planes of Fig. 5.9, we find some similarities of association in all presented quality indices

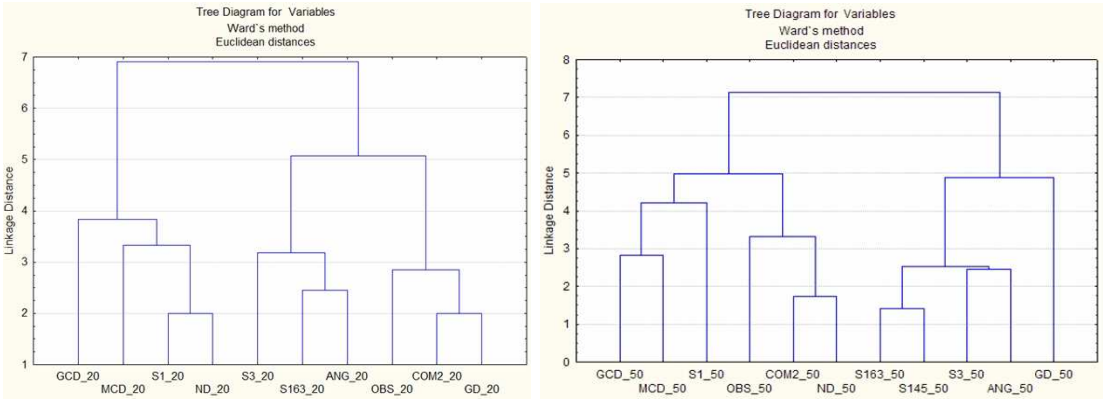



Figure 5.11: Dendograms (left, severe simplification; right, moderate simplification) showing the association between quality results obtained using the observer study and quality indices.

**except** for ANG and GCD. Note that the ND index exhibits a weaker association. On the other hand, by comparing the factorial plane on the right of Fig. 5.4 with those of Fig. 5.10, obtained for moderate simplifications, some similar associations are observed for the ND, GCD and COM2 indices.

These comparison results suggest that, for example, the GD index might be a good estimator of user perceived quality for severe simplifications, but failing for moderate simplifications where the ND or GCD indices seem more appropriate. These results are similar to those obtained in the previous study for lung models. The COM2 index seems to be a good estimator of user perceived quality for both simplification levels ranking all methods the same way users did. It was an expected result since it blends both GD and ND indices, assigning a larger weight to ND when models are moderately simplified and to GD when in the presence of severe simplifications. To confirm these findings Cluster Analysis was once again used. The results obtained in the observer study were treated like a quality index and dendograms were drawn, depicting the associations between all indices. For the severe simplifications (Fig. 5.11, on the left) the observer (OBS) results appear associated with COM2 and GD; for the moderate simplifications (Fig. 5.11, on the right) the observer results appear associated with COM2 and ND.

### 5.1.4 Conclusions

Three mesh simplification methods were compared at two simplification levels. This comparison was performed through two different approaches: an observer study and objective quality indices. While the former approach has the advantage of being capable of producing a *golden standard*, it is very expensive in terms of several resources. That is why it would be very interesting to identify quality indices capable of reasonably estimating the observers' behavior. Yet, quality indices usually don't correlate well with the results obtained through observer studies.

Several quality indices were selected in order to assess if any of them might be such a good estimator of the observers' behavior, at least in some specific circumstances. We confirmed a

result previously presented in Sousa Santos et al. [143] suggesting the Geometric Deviation as a good estimator for severe simplifications and the Normal Deviation as a good estimator for moderate simplifications. This fact had already been taken into account when developing the Mixed Measure quality index (COM2), which was the best estimator for both simplification levels.

Note that the smoothness indices, computed using several neighborhoods, did not provide useful results: only S1 seemed to estimate user performance for severe simplifications. Among the curvature deviations, the Gaussian Curvature Deviation seemed a good estimator of user performance for moderate simplifications. Angle Analysis did not provide any significant results.

Clearly these results do require further confirmation. The results provided by the Mixed Measure were obtained for two simplification levels deemed representative, but how will it behave for a wider range of simplification levels? To answer this question further work needs to be performed in order to understand how the observer perceived quality varies across simplification levels and which adjustments are needed in the Mixed Measure index (e.g., the blending factor) to encompass such changes.

## 5.2 Case Study 2 – Analysis and Comparison of Simplification Algorithms

A task which can be performed using PolyMeCo is the comparison among different simplification methods in order to identify the one which provides meshes of greater quality. In the work described next, the visualization capabilities of PolyMeCo were used in order to compare between two simplification methods.[2]

### 5.2.1 Compared Simplification Methods

*QSlim* [48, 49] and a new algorithm, called *NSA* [133], were compared. Both algorithms simplify a mesh by iteratively collapsing edges into vertices, i.e., using the edge collapse operation.

The edge collapsing operation is standard. The main difference between the various edge collapsing-based simplification algorithms is the criterion used to choose the next edge to collapse. A different criterion implies different mesh quality, as well as a distinct processing time. Generally, all simplification algorithms make a trade-off between speed and the quality of the resulting mesh.

*QSlim* follows a geometric criterion that is based on the minimization of the error associated with each new vertex. This error is defined as the sum of the squared distances to the set of planes surrounding the pair of the original collapsing vertices. Thus, this algorithm produces simplified meshes with a very good geometric quality since it minimizes the error associated with each new vertex.

[2]The work described in this section has been published in Silva et al. [136].

Figure 5.12: Models used for the evaluation. From left to right and from top to bottom: RockerArm, FanDisk, Flasche and BlockFS.

On the other hand, *NSA* algorithm follows a geometric criterion which implies that the region around the collapsing edge is nearly coplanar. An edge is only collapsed if the variation of the face normals around the target edge is within a given tolerance $\varepsilon$. The value of $\varepsilon$ is the threshold for the angle between the current normal and the new normal after the edge collapsing operation.

*NSA* is faster than *QSlim* without losing too much mesh quality [133]. In general, it provides good results in terms of shape preservation, time performance, and mesh quality. Note that *NSA* was primarily developed for time efficiency purposes and mesh quality was not a priority goal.

### 5.2.2 Methodology

For the evaluation of the above mentioned simplification methods four polygonal models were chosen. Figure 5.12 and Table 5.5 show those models and provide some details about their complexity.

For each model and for three simplification levels (strong, moderate and light), a simplified model was created using *QSlim* and another using *NSA*. Figure 5.13 shows the complete set for the Fandisk model.

| Models | # Vertices | # Faces |
|---|---|---|
| FANDISK | 6,475 | 12,946 |
| ROCKERARM | 40,177 | 80,354 |
| FLASCHE | 42,762 | 85,524 |
| BLOCKFS | 12,773 | 25,542 |

Table 5.5: Some details regarding the models used to compare between both simplification methods.

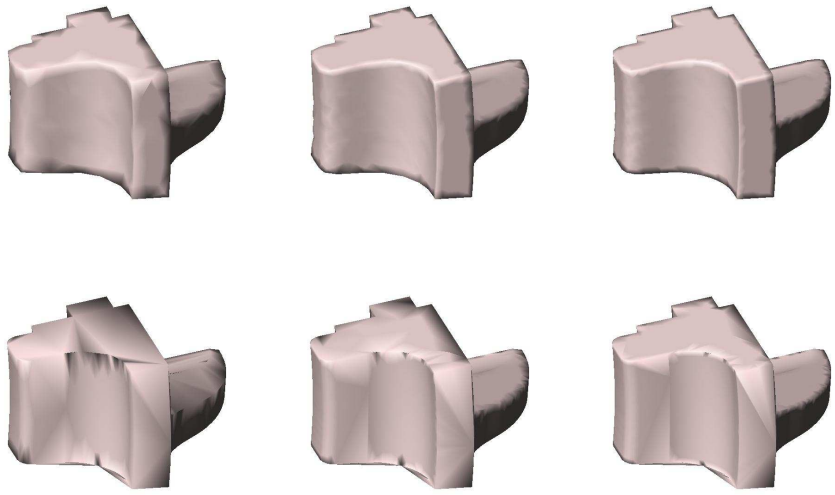


Figure 5.13: Simplified versions for the FANDISK model. Top, strongly, moderately and lightly simplified versions obtained with *NSA*; bottom, the corresponding versions obtained with *QSlim*.

The following results were obtained using POLYMECO.

### 5.2.3 Preliminary Analysis

First, a preliminary analysis was performed regarding the bounding box features and the surface area of the simplified models, comparing them with those of the original models. Specifically, for the FANDISK and BLOCKFS models some important surface area differences were found for the models simplified using *QSlim*, which are related to some surface anomalies that will be analysed later.

A comparative visual analysis of model volume was also performed by rendering the original model and each simplified version simultaneously (i.e., superimposed). Figure 5.14 shows this for two simplified versions of ROCKERARM. The original model is rendered in blue (with the option of changing its transparency level) and the simplified version in pink, thus allowing a perception of the areas where the surface of one of the models is over the other. For all simplified models no significant volume differences were found.

Figure 5.15 shows the superposition of the FANDISK model (with a semi-transparent surface) and one of its simplified versions. On the detail presented on the right it is possible to

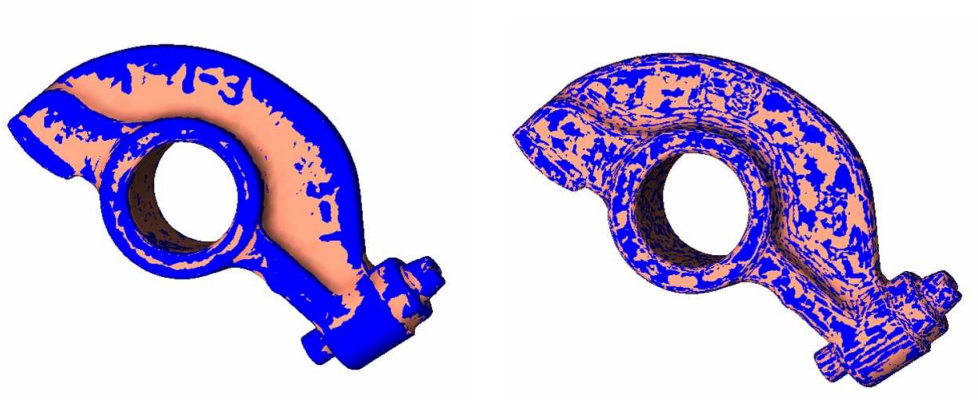


Figure 5.14: Comparative visual analysis of volume differences between the original model and the simplified versions RockerArm_5908_NSA (left) and RockerArm_5908_QS (right). Blue areas correspond to the RockerArm model and pink areas to the simplified model.

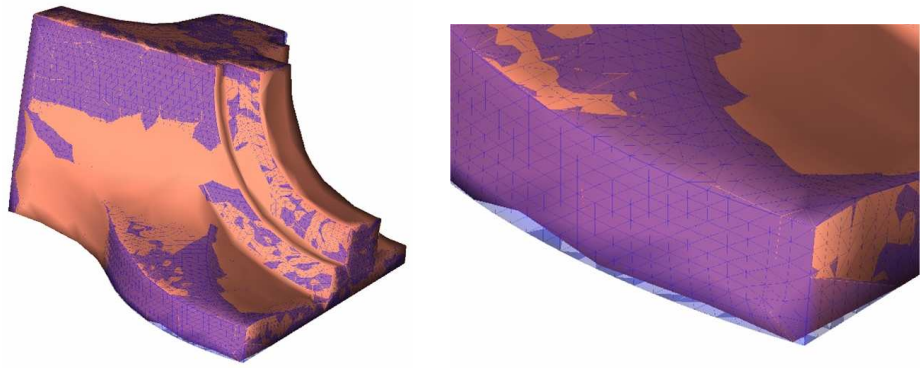


Figure 5.15: Comparative visual analysis using model superposition and setting a degree of transparency for the original model's surface.

detect some minor volume differences.

Proper illumination can also help users analyse a particular mesh and this has been used in very sophisticated ways in the automobile industry [146]. As a first step towards this kind of analysis, PolyMeCo supports adding more point light sources and changing their properties (position and color of components), thus allowing the examination of surfaces in more detail. This kind of analysis is more effective when interactively repositioning the light sources which can also be done in PolyMeCo (although with some limitations depending on the complexity of the viewed mesh). In Fig. 5.16 some of the obtained views are presented; notice, on the top, how several artifacts are noticeable on the surface of the version simplified with *QSlim* and how, on the bottom, the surface highlights look better defined in the model simplified with *QSlim*.

### 5.2.4 Analysis and Comparison using Computational Measures

From the set of computational measures available in PolyMeCo the following were chosen for this case study: Geometric Distance, Normal Deviation, Mixed Measure, Smoothness

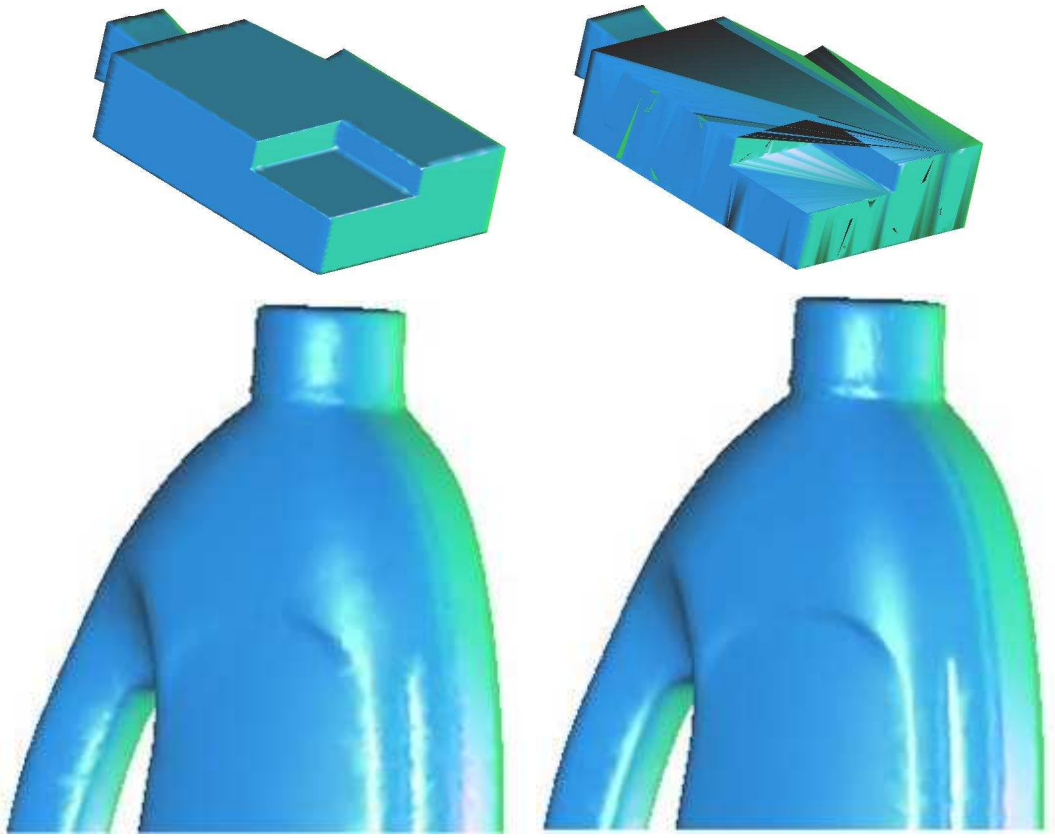


Figure 5.16: Models viewed after customising light source properties. Top, light simplifications of the Blockfs model obtained with NSA (left) and QSlim (right); bottom, details of the moderate simplified versions of Flasche obtained with NSA (left) and QSlim (right).

Analysis and Minimum Angle Analysis. In Table 5.6 we present the mean values obtained for the different metrics when applied to the simplified versions of the chosen meshes. Note that each mesh name includes the number of faces and the used simplification algorithm. For example, Fandisk_7000_NSA is the simplified mesh of FANDISK with 7000 faces generated by *NSA* and Fandisk_7000_QS is the simplified mesh of FANDISK with 7000 faces generated by *QSlim*.

For the FANDISK model in Table 5.6, we can see that *NSA* produces better results than the *QSlim* algorithm for all the metrics with the exception of Geometric Deviation and Angle Analysis for light simplifications. Figure 5.17 shows some models colored according to the results obtained using the Angle Analysis metric. Notice how the bottom models (created using *QSlim*) exhibit a larger amount of irregular triangles (with at least one small angle) which results, for example, in very odd triangulations at the base of the model.

For the ROCKERARM model a different behavior is observed. *QSlim* produces better results than *NSA* for all the metrics with the exception of Angle Analysis and Normal Deviation for light simplification. Figure 5.18 shows some models colored according to the values obtained using the Geometric Deviation. It is clear that the model created using *NSA* exhibits larger deviation values. The shown histograms are an alternative way of looking at these

Table 5.6: Evaluating the simplified models created by *QSlim* and *NSA* algorithms using different metrics. The mean values are presented.

| **Meshes** | **Geom. Dist.** | **Normal Dev.** | **Mixed Measure** | **Smoothness** | **Angle Analysis** |
|---|---|---|---|---|---|
| Fandisk_7000_NSA | 1.65E-4 | 3.44E-2 | 4.66E-2 | 1.47E-5 | 36.72 |
| Fandisk_7000_QSlim | 8.32E-7 | 2.08E-1 | 9.74E-2 | 1.61E-3 | 38.88 |
| Fandisk_3872_NSA | 3.65E-4 | 6.21E-2 | 5.59E-2 | 6.23E-5 | 34.81 |
| Fandisk_3872_QS | 9.83E-6 | 2.60E-1 | 1.37E-1 | 2.51E-3 | 31.77 |
| Fandisk_1312_NSA | 1.22E-3 | 1.28E-1 | 8.72E-2 | 1.61E-4 | 30.89 |
| Fandisk_1312_QS | 4.75E-5 | 3.91E-1 | 9.50E-2 | 3.84E-3 | 20.78 |
| RockerArm_40568_NSA | 1.19E-2 | 2.95E-2 | 9.16E-2 | 2.84E-4 | 35.10 |
| RockerArm_40568_QS | 1.93E-3 | 3.68E-1 | 6.99E-2 | 2.22E-4 | 36.60 |
| RockerArm_20742_NSA | 2.35E-2 | 4.81E-2 | 1.03E-1 | 5.79E-4 | 30.70 |
| RockerArm_20742_QS | 4.76E-3 | 2.81E-2 | 1.02E-1 | 5.15E-3 | 34.60 |
| RockerArm_5908_NSA | 6.59E-2 | 1.00E-1 | 1.48E-1 | 3.81E-3 | 29.70 |
| RockerArm_5908_QS | 1.12E-2 | 6.13E-2 | 1.05E-1 | 1.60E-3 | 28.10 |
| Flasche_46444_NSA | 3.41E-2 | 3.85E-2 | 1.03E-1 | 2.45E-3 | 37.63 |
| Flasche_46444_QS | 8.18E-3 | 3.17E-2 | 5.63E-2 | 6.54E-4 | 38.67 |
| Flasche_25615_NSA | 6.32E-2 | 6.05E-2 | 1.32E-1 | 4.66E-3 | 36.66 |
| Flasche_25615_QS | 1.52E-2 | 4.65E-2 | 8.75E-2 | 1.06E-3 | 32.56 |
| Flasche_3094_NSA | 3.70E-1 | 1.56E-1 | 1.25E-1 | 3.10E-2 | 32.19 |
| Flasche_3094_QS | 4.61E-2 | 1.10E-1 | 8.94E-2 | 2.71E-2 | 24.72 |
| BlockFS_13040_NSA | 3.23E-4 | 3.50E-2 | 2.93E-2 | 2.40E-4 | 26.04 |
| BlockFS_13040_QS | 0.00 | 2.19E-1 | 6.15E-2 | 9.01E-2 | 19.65 |
| BlockFS_7000_NSA | 1.16E-3 | 6.05E-2 | 2.36E-2 | 5.10E-3 | 23.44 |
| BlockFS_7000_QS | 1.00E-6 | 3.17E-1 | 5.31E-2 | 9.80E-2 | 16.31 |
| BlockFS_3684_NSA | 2.75E-3 | 9.52E-2 | 2.66E-2 | 1.55E-3 | 24.99 |
| BlockFS_3684_QS | 1.00E-6 | 3.41E-1 | 3.69E-2 | 1.65E-1 | 18.23 |

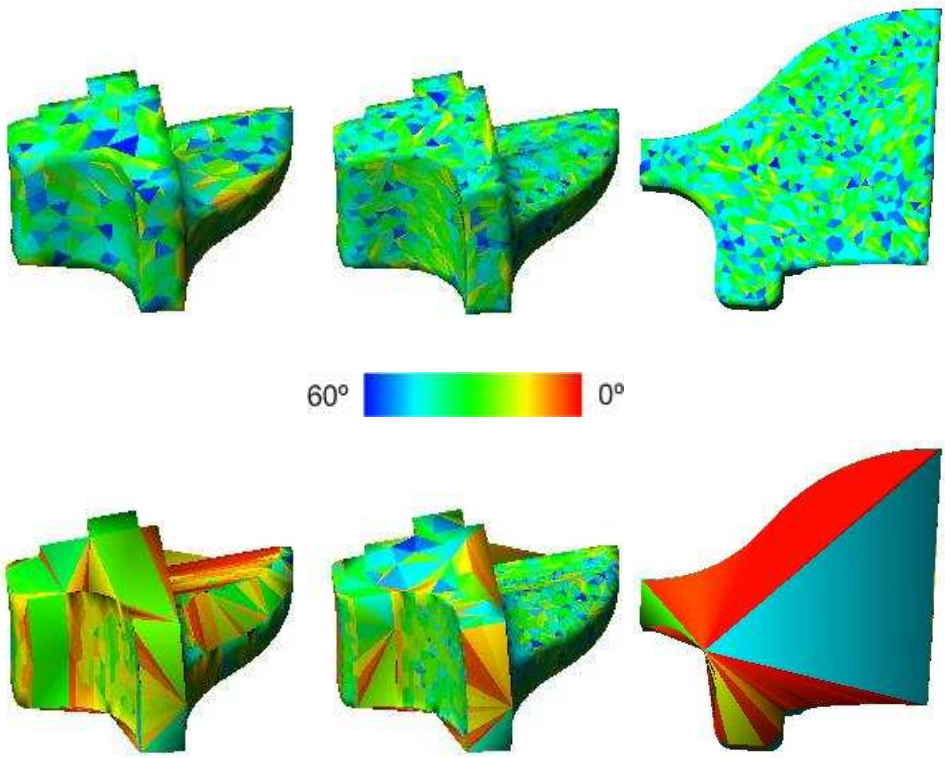



Figure 5.17: Colored models depicting the distribution of the values obtained with the Angle Analysis metric for the FANDISK model. Top, strongly, moderately and lightly simplified versions obtained with *NSA*; bottom, the corresponding versions obtained with *QSlim*. Areas in red are composed of flat or needle triangles.

results. Figure 5.19 shows a detail of the ROCKERARM model when evaluated using the Composed Deviation. Once again the model created using *NSA* obtained poorer results.

*QSlim* also produces better results for the FLASCHE model (Table 5.6) in all metrics except for the Angle Analysis regarding the light simplifications. Figure 5.20, on the left, shows two models colored according to the computed Normal Deviation. The model created with *NSA* exhibits a larger deviation towards the original; on the right it is possible to verify a poorer triangle quality (smaller angles) in the model created with *QSlim*.

Finally, for the BLOCKFS model, *NSA* produces better results with the exception of those obtained using Geometric Deviation. Figure 5.21 shows models colored (using a common color map) according to the Normal Deviation for all of the simplified models. It is clear that all models created with *QSlim* exhibit larger deviation, mainly due to some triangulation problems similar to those shown in Fig. 5.17 for the FANDISK model. Figure 5.22 shows one of POLYMECO's visualization modes, *Extended Results*, showing the results obtained for model RockerArm_40568 using the Mixed Measure. By observing the boxplot, on the top right corner, it is possible to verify the existence of several points (in blue) above the top whisker. These points represent outliers detected on the data. The color scale was adjusted (by using the sliders below the histogram) in order to exclude those values, which enables a

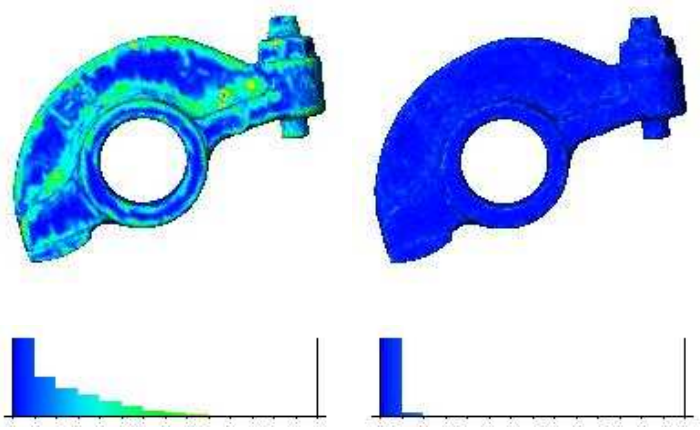


Figure 5.18: Colored models and histograms depicting the distribution obtained using the Geometric Distance for the RockerArm_5908_NSA (left) and RockerArm_5908_QS (right) models.

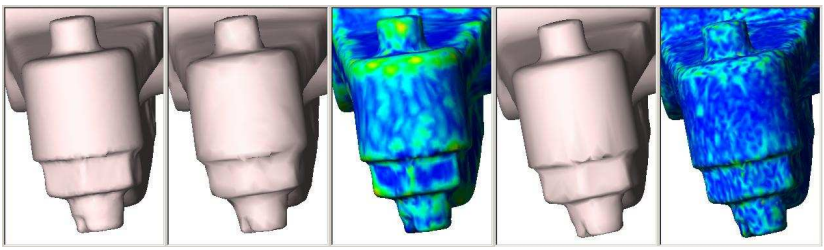


Figure 5.19: Composed Deviation results for the RockerArm model seen in greater detail. From left to right: original model, RockerArm_5908_NSA and results, and RockerArm_-5908_QS and results.

better visualization of the results (Fig. 5.22(b)).

### 5.2.5 Conclusions

From the analysis of the obtained results it is possible to conclude that *QSlim* always leads to simplified meshes with a lower mean Geometric Deviation when compared with the original mesh.

For the other metrics results depend on the nature of the mesh. For example, when dealing with meshes with planar regions (Fandisk and BlockFS) *NSA* produces better results than *QSlim* for all metrics except Geometric Deviation.

For models without planar regions *QSlim* produces, in general, better results for all metrics. However, for the same meshes, *NSA* yields better Minimum Angle Analysis results, as for the RockerArm model. For models with planar regions *QSlim* creates some surface artifacts due to the existence of vertices with a large number of incident edges, which explains why it normally yielded worst results for the Minimum Angle Analysis. This particularity can be a problem for visualization purposes and for finite element simulations where triangle

Figure 5.20: On the left, models colored using a blue-to-cyan colorscale according to the Normal Deviation results computed for models Flasche_3094_NSA and Flasche_3094_QS. On the right, Angle Analysis results for these same models.

quality is of extreme importance.

In fact, meshes created by *QSlim* have less geometric error than meshes created by *NSA*, but their visual quality is poorer when compared with the quality of the meshes created by *NSA*, in particular for models with planar regions.

PolyMeCo provided the possibility of analysing several simplified meshes using a large range of metrics, which allowed a more precise evaluation of the simplification algorithms. The *Features Comparison* visualization mode was particularly useful since it allowed comparisons (which is a harder task to perform when using just numerical values) using colored models and histograms (with common color maps and value ranges). The available visualization options allowed a clearer understanding of the meaning of numerical values as well as the problems they are related to. A clear example is that of Angle Analysis values and the detection of odd triangles.

Having all models and data (from all the computed measures) simultaneously available in PolyMeCo allowed a more interactive analysis: it was possible to change between models and visualization modes while searching for the configurations which provided greater insight into the analysed data.

These features allowed a better understanding of the main characteristics of each algorithm. For instance, if the original model has a large number of planar regions, and visual quality is the most important criterion, *QSlim* might not be the algorithm of choice.

## 5.3 Conclusion

This chapter presents two application examples for PolyMeCo. On the first example, PolyMeCo was used in a more conservative fashion, just like other tools (e.g., Metro [37]) have been used in the literature, to compute quality indices for different polygonal meshes obtained with three simplification methods. The main goal was to search for good estimators of user perceived quality, obtained using an observer study, in order to establish guidelines which allow users to more clearly understand the information provided by quality indices.

In the second example presented, the visualization features provided by *PolyMeCo* where used to compare between two simplification methods. This simultaneous numerical and visual

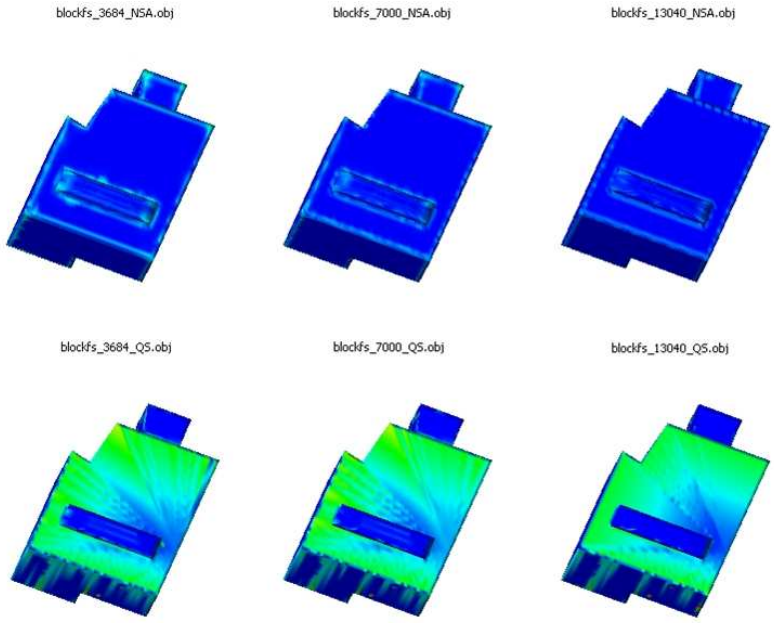
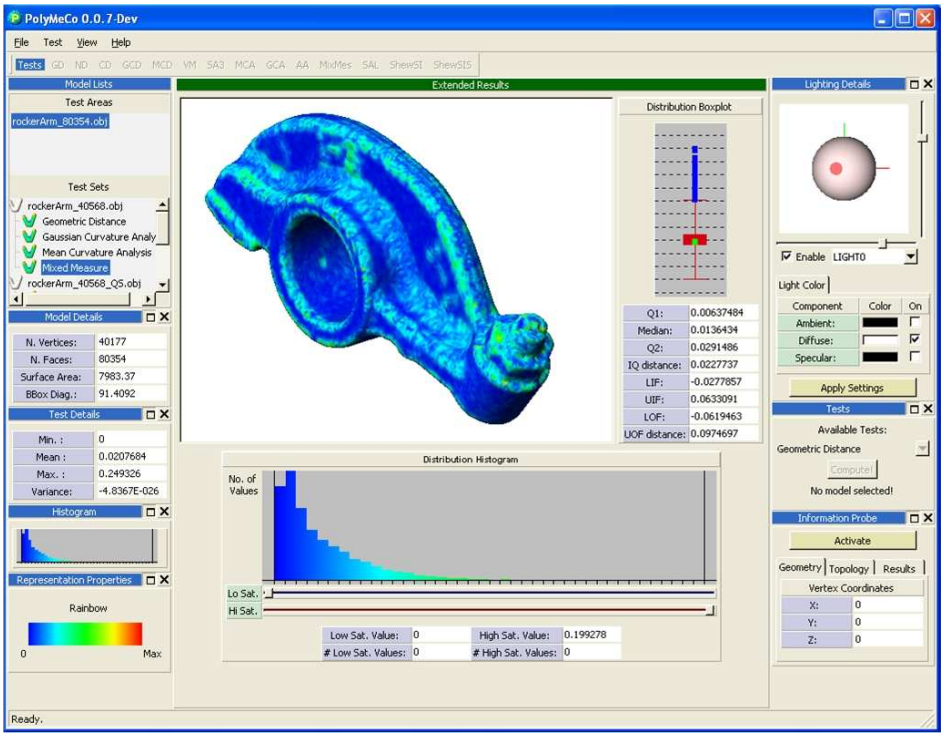


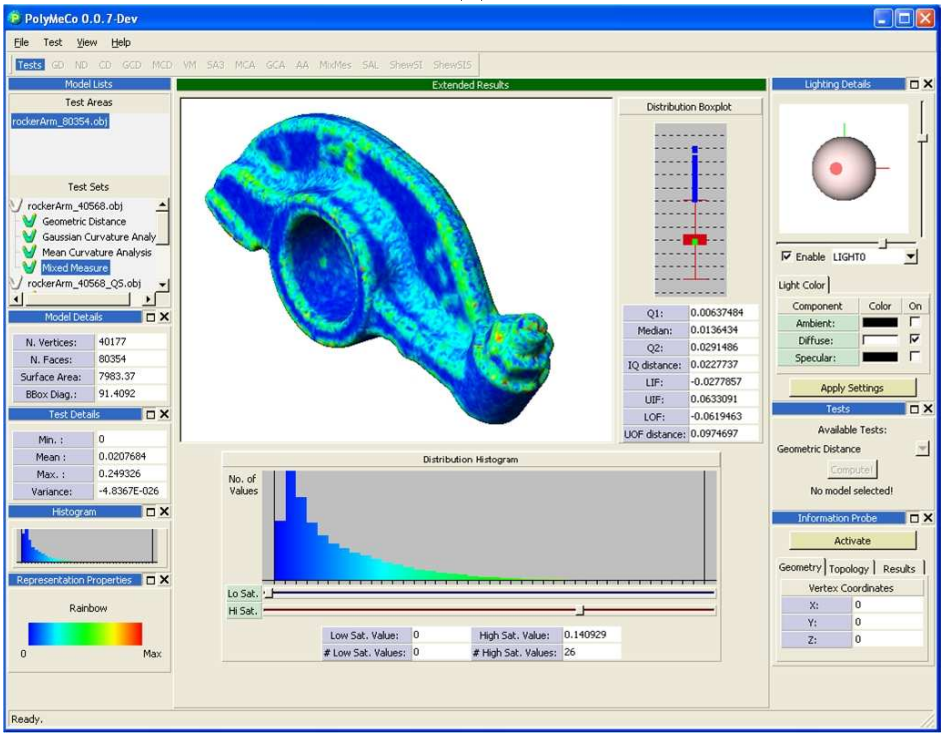

Figure 5.21: Colored models depicting the results obtained for the simplified versions of BlockFS using the Normal Deviation metric. A common color map is used for all models thus allowing direct results comparison using the colored models. On the top, strongly, moderately and lightly simplified models using *NSA*, on the bottom, the correspondent simplified models using *QSlim*.

analysis and comparison of polygonal meshes helps users to understand how numerical values obtained with several measures express what is happening visually on the models.

These two application examples helped evaluating PolyMeCo to detect bugs, tune features and resulted in new ideas for future development.




(a)




(b)

Figure 5.22: Extended Results *Visualization Mode* for model RockerArm_40568 showing a colored model, a histogram and a boxplot for the results obtained with the Mixed Measure: (a) default color mapping; (b) the color mapping was adjusted (trying to eliminate the outliers) resulting in a clearer visualization of the results.

# Chapter 6

# Conclusions and Further Work

In the scope of the work carried out for this dissertation, PolyMeCo, a tool for polygonal mesh analysis and comparison, was designed, developed and tested. After an introduction stating the motivation and objectives, short overviews on Data Visualization, focusing on important aspects regarding the developed work, and on Geometric Modeling using polygonal meshes, describing some of the usual mesh processing operations along with a set of methods and tools to analyse and compare polygonal meshes, followed.

Having considered that existing tools for mesh analysis and comparison lack several important features, the new tool, PolyMeCo, was presented describing its purpose, architecture and features. This tool was then used in two research activities presented as application examples.

In the following sections a summary of the most important features provided by PolyMeCo and some ideas for further work will be presented.

## 6.1 Conclusions

PolyMeCo has several innovative features that distinguish it from similar tools presented in the literature:

1. Integrated environment where several models can be analyzed/compared simultaneously;
2. Several data representations which are arranged in visualization modes to provide different alternative visualizations of the data;
3. Customization of light source properties;
4. Location probe to obtain specific information about the computed data;
5. Enabled/disabled model rendering mode to speed up some tasks;
6. Workspace saving.

The provided integrated environment allows working with several models simultaneously, thus speeding the analysis and comparison process. After loading all the models, the user can easily compute several measures and then analyse the obtained data. An important feature is that no third-party application is needed to visualize the data as it happens with other tools, such as MeshDev [123].

Data obtained using the provided *Computational Measures* can be presented to the user through one of the available *Representations* (e.g., using color). For this purpose, several color scales are available in order to allow a proper choice, as judged by the user. While working with *Representations* using color, the user can adjust the color mapping when in the presence of data containing outliers (in order to eliminate them from the color mapping) or to highlight specific values.

*Representations* are presented as part of *Visualization Modes*. These articulate *Representations* in order to build proper visualizations of the data. Among those available in PolyMeCo the Feature Comparison *Visualization Mode* is of paramount importance, as it allows comparing models using color *Representations* (or histograms). This is performed by allowing the definition of a common color mapping among all models. Without this innovative feature colored models can only be used to compare *Computational Measures* incidence among models, but never their relative values, as it is performed in [166].

To enhance the way the model's surface can be inspected, the user can customize the light properties of the scene by adding up to three light sources and controlling their position and ambient, diffuse and specular components. Customizing the light properties can help to detect visual artifacts in the model's surface, as was shown in Chapter 5.

By providing a location probe, PolyMeCo allows the user to obtain information regarding vertices, namely its coordinates, neighborhood and associated value according to one *Computational Measure*, which is a feature not available in any of the tools described in the literature.

When working with very complex models (i.e., with a large mumber of vertices and faces) it can sometimes be difficult to visualize them due to the associated rendering times: for example, when just selecting each of the loaded models to get information about their properties (e.g., surface area, bounding box diagonal) or computed data (Section 5.1 presents a good example). For these particular situations PolyMeCo provides an offline working mode in which model rendering is disabled. As a consequence, this also disables some important visualization features of PolyMeCo but allows accomplishing the other tasks much faster. The user can enable/disable model rendering at any time.

Performing mesh analysis and comparison may require storing the computed data for future reference or to continue the analysis later. As some *Computational Measures* take quite some time to compute, it would be a tedious task to compute them whenever they were necessary. For this reason, PolyMeCo provides saving the current workspace (i.e., all loaded models and computed measures) and restore it later. It is also possible to export the data obtained using any *Computational Measure* to allow, for example, further analysis using a different tool.

The presented application examples have shown that PolyMeCo is, indeed, a suitable tool in its field. Users have expressed their appreciation towards PolyMeCo stating that one of its advantages is the fact that it allows experimenting with the different *Computational Measures* and *Visualization Modes* in a very easy way.

Throughout the dissertation, features were usually illustrated using simplification methods. This is only due to the fact that they are good representatives of mesh processing methods in general and they are easily accessible. PolyMeCo can be used to analyse and compare polygonal mesh models in general, regardless of the processing to which they were submitted. An example of such versatility is the way PolyMeCo has been used to compare three skinning frameworks for character animation as described in Reid et al. [111].

## 6.2 Further Work

All the above described characteristics establish PolyMeCo as a tool for systematic mesh analysis and comparison, not only for mesh processing algorithm developers but also for those who perform mesh processing for particular purposes. Still, it can be enhanced in many ways.

New *Computational Measures* may be added. All the mesh comparisons performed in PolyMeCo consider that the compared models have their principal axes aligned and that all differences are due to surface distortion. Therefore, a model compared with a version of itself rotated/translated in space will result in significant differences, e.g., large Geometric Distances when the difference is only due to affine transformations. These differences must also be reported to the user, but stating their nature. To deal with this it might be a good idea to use methods such as Principal Component Analysis [106] and the Extended Gaussian Image [64, 144]. It would also be interesting to test if 3D shape descriptors [30, 112], usually used in retrieving models from databases, can be used to compare polygonal meshes. Another interesting feature to add might be that of spectral analysis of mesh surfaces [102, 140], in a similar approach to that provided by Fourier Analysis regarding, for example, audio signals.

Although several color scales are provided, it is up to the user to select the one he/she thinks most appropriate for the data being visualized. It would be desirable that PolyMeCo provide a suggestion based upon the principles presented in Chapter 2. This requires methods to measure spatial frequencies. It must not be forgotten that when coloring a 3D model, according to a certain data set, two things are changing which affect spatial frequency: the data and the model's surface. PolyMeCo might also suggest a proper adjustment of the color mapping when in the presence of outliers on the data.

The light source properties customization can be enhanced in order to allow surface examination using isophotes [27]. In particular, the light sources must be modeled differently to obtain the desired effects on the surface.

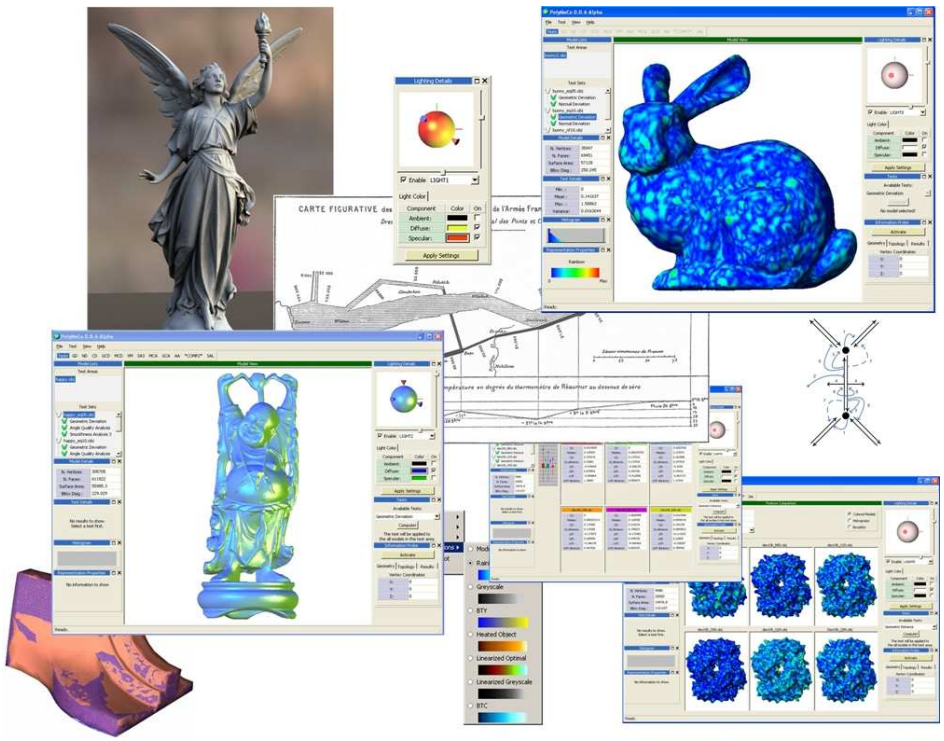
The Location Probe can be improved in order to provide information regarding all the loaded models (not only the models depicting analysis/comparison data). It would also be helpful that, for example, while in the Feature Comparison *Visualization Mode*, the Location Probe could be used, in parallel, on all the models, i.e., selecting a vertex in one of the

models would also give information about the corresponding vertex in all the others. This, of course, would not be possible for all mesh processing methods as it depends on a vertex-to-vertex correspondence between models. The Location Probe might also support obtaining information about mesh faces which would be helpful, for example, when visualizing data obtained with the Triangle Quality *Computational Measure*.

Workspace saving is still performed in a very simple way. There are several improvements which can be applied on the file format used, e.g., regarding the storage of the data obtained using the *Computational Measures*. There are also some desirable features like selective loading/saving of only part of the information contained in the workspace, or a list of style sheets to associate with the XML file to provide different views of the data, for example, in a web browser.

It might be useful to provide some simple mesh editing tools in PolyMeCo to modify, for example, the position of a vertex. This could help the user correct some minor problems on a mesh.

Finally, a thorough analysis must be performed to understand if it would be helpful to provide a module which allows mesh processing methods integration: instead of creating the models to be compared outside PolyMeCo, they would be created inside the integrated environment.



# Bibliography


[1] Cortona VRML client. In *http://www.parallelgraphics.com/products/cortona/*, (online December 2006).

[2] eXtensible Markup Language (XML). In *http://www.w3.org/XML/*, (online December 2006).

[3] Fox toolkit. In *http://www.fox-toolkit.org*, (online December 2006).

[4] Iris Explorer. In *http://www.nag.co.uk/welcome_iec.asp*, (online December 2006).

[5] OBJ file format. In *http://www.fileformat.info/format/wavefrontobj/*, (online December 2006).

[6] OpenMesh library. In *http://www.openmesh.org*, (online December 2006).

[7] PLY file format. In *http://local.wasp.uwa.edu.au/~pbourke/dataformats/ply/*, (online December 2006).

[8] Qt. In *http://www.trolltech.com/products/qt/*, (online December 2006).

[9] Stanford scanning repository. In *http://graphics.stanford.edu/data/3Dscanrep*, (online December 2006).

[10] wxWidgets. In *http://www.wxwidgets.org/*, (online December 2006).

[11] G. Agam and X. Tang. A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 11(5): 573–583, 2005.

[12] P. Alliez and C. Gotsman. Recent advances in compression of 3D meshes. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometric Modeling*, pages 3–26. Springer-Verlag, 2005.

[13] E. Angel. *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*. Addison Wesley, 3rd edition, 2003.

[14] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. MESH: Measuring errors between surfaces using the Hausdorff distance. In *Proc. IEEE International Conference in Multimedia and Expo 2002*, volume 1, pages 705–708, Lausanne, Switzerland, 2002.

[15] D. Azuma, D. Wood, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle. View-dependent refinement of multiresolution meshes with subdivision connectivity. In *Proc. AFRIGRAPH 2003: 2nd International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction, in Africa*, pages 69–78, Cape Town, South Africa, 2003.

[16] V. Barnett. *Sample Survey Principles and Methods.* Arnold Hodder, 3rd edition, 2003.

[17] D. Bartz. Virtual endoscopy in research and clinical practice. *Computer Graphics Forum*, 24(1):111–137, 2005.

[18] B. Baumgart. A polyhedron representation for computer vision. In *Proc. National Computer Conference*, pages 589–596, Anaheim, California, 1975.

[19] O. Benedens. Robust watermarking and affine registration for 3D meshes. *Lecture Notes in Computer Science*, 2578:177–195, 2003.

[20] L. Bergman, B. Rogowitz, and L. Treimish. A rule-based tool for assisting color map selection. In *Proc. IEEE Visualization '95*, pages 118–125, 1995.

[21] I. Biederman and P. Kalocsai. Neural and psychological analysis of object and face recognition. In *Face Recognition: From Theory to Applications*, pages 3–25. Springer-Verlag, 1998.

[22] S. Bischoff and L. Kobbelt. Teaching meshes, subdivision and multiresolution. *Computer-Aided Design*, 36(14):1483–1500, 2004.

[23] CGAL Editorial Board. *CGAL-3.2 User and Reference Manual.* 2006.

[24] Y. Boon-Lock and M. Minerva. Watermarking 3D objects for verification. *IEEE Computer Graphics & Applications*, 19(1):36–45, 1999.

[25] D. Borland and R. Taylor. Rainbow color map (still) considered harmful. *IEEE Computer Graphics & Applications*, 27(2):14–17, 2007.

[26] A. Bors. Watermarking mesh-based representations of 3D objects using local moments. *IEEE Transactions on Image Processing*, 15(3):687–701, 2006.

[27] M. Botsch, M. Pauly, C. Rössl, S. Bischoff, and L. Kobbelt. Geometric modeling based on triangle meshes. In *EUROGRAPHICS 2006 – Tutorials*, Vienna, Austria, 2006.

[28] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. OpenMesh - a generic and efficient polygon mesh data structure. In *1st OpenSG Symposium*, Darmstadt, Germany, 2002.

[29] K. Brodlie, L. Carpenter, R. Earnshaw, J. Gallop, R. Hubbold, A. Mumford, C. Osland, and P. Quarendon. *Scientific Visualization, Techniques and Applications.* Springer, 1992.

[30] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. An experimental effectiveness comparison of methods for 3D similarity search. *International Journal on Digital Libraries*, 6(1):39–54, 2006.

[31] S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges — a scalable representation for triangular meshes. *Journal of Graphics Tools*, 3(4):1–12, 1998.

[32] W. Cao. On the error of linear interpolation and the orientation, aspect ratio and internal angles of a triangle. *SIAM Journal on Numerical Analysis*, 43(1):19–40, 2005.

[33] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think.* Morgan Kauffman Publishers, San Francisco, 1999.

[34] C.-Y. Chen and K.-Y. Cheng. A sharpness dependent filter for mesh smoothing. *Computer-Aided Geometric Design*, 22(5):376–391, 2005.

[35] J.-J. Chen, W. Liu, M.-Z. Li, and C.-T. Wang. Digital manufacture of titanium prosthesis for cranioplasty. *International Journal of Advanced Manufacturing Technology*, 27:1148–1152, 2006.

[36] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[37] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[38] P. Cignoni, R. Scopigno, and M. Tarini. A simple normal enhancement technique for interactive non-photorealistic renderings. *Computers & Graphics*, 29(1):125–133, 2005.

[39] W. Cleveland and R. McGill. A color-caused optical ilusion on a statistical graph. *The American Statistician*, 37(2):101–105, 1983.

[40] W. Cochran. *Planning and Analysis of Observational Studies.* John Wiley, 1983.

[41] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proc. SIGGRAPH 1998*, pages 115–122, 1998.

[42] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, 2004.

[43] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. SIGGRAPH 1999*, pages 317–324, 1999.

[44] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics*, 22:950–953, 2003.

[45] L. Floriani and A. Hui. Data structures for simplicial complexes: an analysis and comparison. In *Proc. 3rd Eurographics Symposium on Geometry Processing*, pages 119–128, Vienna, Austria, 2005.

[46] M. Frigge, D. Hoaglin, and B. Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, 1989.

[47] R. Gandhi, G. Kumar, B. Bederson, and B. Shneiderman. Domain name based visualization of web histories in a zoomable user interface. In *Proc. 11th International Workshop on Database and Expert Systems Applications*, pages 591–598, 2000.

[48] M. Garland. QSlim. *http://www.cs.cmu.edu/~garland*, (online December 2006).

[49] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH 1997*, pages 209–216, 1997.

[50] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004.

[51] M. Gross, O. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):130–144, 1996.

[52] X. Gu. Harvard graphics archive – mesh library. In *http://people.deas.harvard.edu/~xgu/mesh/*, (online December 2006).

[53] G. Guidi, J. Beraldin, and C. Atzeni. High-accuracy 3-D modeling of cultural heritage: The digitizing of Donatello's "Maddalena". *IEEE Transactions on Image Processing*, 13 (3):370–380, 2004.

[54] G. Guidi, B. Frischer, M. De Simone, A. Cioci, A. Spinetti, L. Carosso, L. Micoli, M. Russo, and T. Grasso. Virtualizing ancient Rome: 3D acquisition and modeling of a large plaster-of-Paris model of imperial Rome. In *Proc. SPIE Vol. 5665, Videometrics VIII*, pages 119–133, 2005.

[55] M. Guthe, P. Borodin, and R. Klein. Fast and accurate Hausdorff distance calculation between meshes. *Journal of WSCG*, 13:41–48, 2005.

[56] B. Hamann. Curvature approximation for triangulated surfaces. *Springer Computer Supplementum - Geometric Modeling*, pages 139–153, 1993.

[57] T. Harte and A. Bors. Watermarking 3D models. In *Proc. International Conference on Image Processing*, pages 661–664, 2002.

[58] C. Healey. Choosing effective colours for data visualization. In *Proc. IEEE Visualization '96*, pages 263–270, 1996.

[59] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. In *Technical Report*, Carnegie-Mellon Univ., School of Computer Science, 1997.

[60] M. Hisada and A. Belyaev. A skeleton-based approach to predicting saliency for simplified polygonal models. *Computer Graphics Forum*, 21(4):689–700, 2002.

[61] T.-C. Ho, Y.-C. Lin, J.-H. Chuang, C.-H. Peng, and Y.-J. Cheng. User-assisted mesh simplification. In *Proc. ACM International Conference on Virtual Reality Continuum and its Applications*, pages 59–66, 2006.

[62] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH 1996*, pages 99–108, 1996.

[63] H. Hoppe. View-dependent refinement of progressive meshes. In *Proc. SIGGRAPH 1997*, pages 189–198, 1997.

[64] B. K. P. Horn. Extended gaussian images. *Proc. of IEEE*, 72(12):1671–1686, 1984.

[65] S. Howlett, J. Hamill, and C. O'Sullivan. Predicting and evaluating saliency for simplified polygonal models. *ACM Transactions on Applied Perception*, 2(3):286–308, 2005.

[66] W. Huang. Measuring mesh qualities and application to variational mesh adaptation. *SIAM - Journal of Scientific Computing*, 26(5):1643–1666, 2005.

[67] K. Inagaki, M. Okuda, M. Ikehara, and S. Takahashi. Measuring errors on 3D meshes using pixel based search. *IEICE Transactions on Information and Systems*, E86-D(9): 1903–1908, 2003.

[68] M. Jackowski, M. Satter, and A. Goshtasby. Approximating digital 3D shapes by rational gaussian surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):56–69, 2003.

[69] J. Jang, P. Wonka, W. Ribarsky, and C. Shaw. Punctuated simplification on man-made objects. *The Visual Computer*, 22(2):136–145, 2006.

[70] D. G. Johnson. *Applied Multivariate Methods for Data Analysis*. Duxbury, 1998.

[71] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature preserving, mesh smoothing. *ACM Transactions on Graphics*, 22:943–949, 2003.

[72] B.-S. Jong, J.-L. Tseng, and W.-H. Yang. An efficient and low-error mesh simplification method based on torsion detection. *The Visual Computer*, 22(1):56–67, 2006.

[73] B. T. Julesz. The elements of texture perception, and their interactions. *Nature*, 290 (12):91–97, 1981.

[74] S. Kanai, H. Date, and T. Kishinami. Digital watermarking of 3D polygons using multiresolution wavelet decomposition. In *Proc. 6th International Workshop on Geometric Modelling: Fundamentals and Applications*, pages 296–307, 1998.

[75] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proc. SIGGRAPH 2000*, pages 279–286, 2000.

[76] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications*, 13(1):65–90, 1999.

[77] Y Kho and M. Garland. User-guided simplification. In *Proc. ACM Symposium on Interactive 3D Graphics*, pages 123–126, 2003.

[78] G. Kidlmann, D. Weinstein, A. Lee, and P. Thompson. Visualization of anatomic covariance tensor fields. In *Proc. 26th Annual International Conference of the Engineering in Medicine and Biology Society*, pages 1842–1845, 2004.

[79] J. Kim and S. Lee. Truly selective refinement of progressive meshes. In *Proc. Graphics Interface 2001*, pages 101–110, 2001.

[80] G. Kindlmann, E. Reinhard, and S. Creem. Face-based luminance matching for perceptual colormap generation. In *Proc. IEEE Visualization '02*, pages 299–306, 2002.

[81] L. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3):142–158, 2000.

[82] L. Kobbelt, S. Bischoff, M. Botsch, K. Kähler, C. Rössl, R. Shneider, and J. Vorsatz. Geometric modeling based on polygonal meshes. In *EUROGRAPHICS 2000 – Tutorials*, Interlaken, Switzerland, 2000.

[83] R. Kosara, C. Healey, V. Interrante, D. Laidlaw, and C. Ware. Visualization viewpoints – user studies: Why, how, and when? *IEEE Computer Graphics & Applications*, 23(4): 20–25, 2003.

[84] C. Lee, A. Varshney, and D. Jacobs. Mesh saliency. In *Proc. SIGGRAPH 2005*, pages 659–666, 2005.

[85] H. Levkowitz and G. Herman. Color scales for image data. *IEEE Computer Graphics & Applications*, 12(1):72–80, 1992.

[86] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proc. SIGGRAPH 2000*, pages 131–144, 2000.

[87] P. Lindstrom. Out-of-core simplification of large polygonal models. In *Proc. SIGGRAPH 2000*, pages 259–262, 2000.

[88] P. Lindstrom and C. T. Silva. A memory insensitive technique for large model simplification. In *Proc. IEEE Visualization '01*, pages 121–126, 2001.

[89] F. Losasso, I. Geoffrey, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3): 343–352, 2006.

[90] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics & Applications*, 21(3):24–35, 2001.

[91] D. Luebke and B. Hallen. Perceptually driven simplification for interactive rendering. In *Proc. 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, 2001.

[92] J. Mackinlay, G. Robertson, and S. Card. The perspective wall: Detail and context smoothly integrated. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pages 173–176, 1991.

[93] E. Magid, O. Soldea, and E. Rivlin. A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *Computer Vision and Image Understanding, (in press)*, 2006.

[94] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Proc. VisMath 2002*, pages 34–57, 2002.

[95] E. Montag. The use of color in multidimensional graphical information display. In *Proc. IS&T/SID 7th Color Imaging Conference*, pages 222–226, 1999.

[96] K. Müller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand. Predictive compression of dynamic 3D meshes. In *Proc. IEEE International Conference on Image Processing (ICIP 05)*, pages 11–14, 2005.

[97] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE Journal on Selected Areas in Communications*, 16(4):551–560, 1998.

[98] R. Ohbuchi, A. Mukaiyama, and S. Takahashi. A frequency domain approach to watermarking 3D shapes. *Computer Graphics Forum*, 21:373–382, 2002.

[99] R. Ohbuchi, S. Takahashi, T. Miyasawam, and A. Mukaiyama. Watermarking 3D polygonal meshes in the mesh spectral domain. In *Proc. Graphics Interface 2001*, pages 9–17, 2001.

[100] Y. Ohtake and A. G. Belyaev. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(11):789–800, 2001.

[101] K. Park, K. Lee, and S. Lee. Efficient measurement of shape dissimilarity between 3D models using Z-buffer and surface roving method. *EURASIP Journal of Applied Signal Processing*, 10:1127–1134, 2002.

[102] M. Pauly and M. Gross. Spectral processing of point-sampled geometry. In *Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 379–386, 2001.

[103] F. Payan and M. Antonini. An efficient bit allocation for compressing normal meshes with an error-driven quantization. *Computer-Aided Geometric Design*, 22(5):466–486, 2005.

[104] P. Pébay and T. Baker. Analysis of triangle quality measures. *Mathematics of Computation*, 72(244):1817–1839, 2003.

[105] J. Peng, C.-S. Kim, and C.-C. Jay Kuo. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication & Image Representation*, 16:688–733, 2005.

[106] M. Petrou and P. Bosdogianni. *Image Processing: The Fundamentals*. John Wiley & Sons, 1999.

[107] S. Pizer, B. Zimmerman, and E. Johnston. Contrast transmission in medical image display. In *Proc. 1st International Symposium on Medical Imaging and Interpretation*, pages 2–9, 1982.

[108] E. Pojar and D. Schmalstieg. User-controlled creation of multiresolution meshes. In *Proc. ACM Symposium on Interactive 3D Graphics*, pages 127–130, 2003.

[109] K. Rajamani, S. Joshi, and M. Styner. Bone model morphing for enhanced surgical visualization. In *Proc. IEEE International Symposium on Biomedical Imaging: Macro to Nano*, pages 1255–1258, 2004.

[110] A. Razdan and M. Bae. Curvature estimation scheme for triangle meshes using biquadric bézier patches. *Computer-Aided Design*, 37:1481–1491, 2005.

[111] A. Reid, D. Jacka, B. Merry, and J. Gain. Mesh approximation for animated characters. In *Technical Report CS06-06-00*, Department of Computer Science, University of Cape Town, 2006.

[112] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-Beltrami spectra as shape-DNA of surfaces and solids. *Computer-Aided Design*, 38:342–366, 2006.

[113] P. Rheingans. Task-based color scale design. In *Proc. SPIE Vol. 3905, 28th AIPR Workshop: 3D Visualization for Data Exploration and Decision Making*, pages 35–43, 2000.

[114] P. Rheingans and B. Tebbs. A tool for dynamic explorations of color mappings. *Computer Graphics*, 24(2):145–146, 1990.

[115] P. Robertson. Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *IEEE Computer Graphics & Applications*, 8(5): 50–63, 1988.

[116] P. Robertson and J. O'Callagham. The generation of color sequences for univariate and bivariate mapping. *IEEE Computer Graphics & Applications*, 6(2):24–32, 1986.

[117] B. Rogowitz and A. Kalvin. The "Which Blair Project'': A quick visual method for evaluating perceptual color maps. In *Proc. IEEE Visualization '01*, pages 21–26, 2001.

[118] B. Rogowitz and L. Treinish. How not to lie with visualization. *Computers in Physics*, 10(3):268–273, 1996.

[119] B. Rogowitz and L. Treinish. Data visualization: The end of the rainbow. *IEEE Spectrum*, 35(12):52–59, 1998.

[120] B. Rogowitz and L. Treinish. Why should engineers and scientists be worried about color? In *http://www.research.ibm.com/people/l/lloydt/color/color.HTM*, (online December 2006).

[121] J. Rossignac. 3D mesh compression. In C. Hansen and C. Johnson, editors, *Visualization Handbook*, pages 359–380. Academic Press, 2004.

[122] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[123] M. Roy, S. Foufou, and F. Truchetet. Mesh comparison using attribute deviation metric. *Internation Journal of Image and Graphics*, 4(1):1–14, 2004.

[124] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proc. of 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 486–493, 2004.

[125] G. Sansoni and F. Docchio. 3-D optical measurements in the field of cultural heritage: The case of the Vittoria Alata of Brescia. *IEEE Transactions on Instrumentation and Measurement*, 54(1):359–368, 2005.

[126] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics.* Kitware Inc., 3rd edition, 2004.

[127] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *Proc. SIGGRAPH 1992*, pages 65–70, 1992.

[128] A. Shamir. Segmentation and shape extraction of 3d boundary meshes. In *EUROGRAPHICS 2006 — State-of-the-Art Report*, Vienna, Austria, 2006.

[129] Y. Shen and K. Barner. Fuzzy vector media-based surface smoothing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):252–265, 2004.

[130] H. Shimazaki and S. Shinomoto. A method for selecting the bin size of a time histogram. *Neural Computation*, (in press), `http://www.ton.scphys.kyoto-u.ac.jp/~hideaki/res/histogram.html`, 2007.

[131] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2*. Addison-Wesley Professional, 5th edition, 2005.

[132] F. Silva and A. Gomes. Adjacency and incidence framework — a data structure for efficient and fast management of multiresolution meshes. In *Proc. GRAPHITE 2003*, pages 159–166, 2003.

[133] F. Silva and A. Gomes. Normal-based simplification algorithm for meshes. In *Proc. of Theory and Practice of Computer Graphics (TPCG'04)*, pages 211–218, 2004.

[134] S. Silva, C. Ferreira, J. Madeira, and B. Sousa Santos. Perceived quality of simplified polygonal meshes: Evaluation using observer studies. In *Proc. Ibero-American Symposium in Computer Graphics SIAG06*, pages 169–178, Santiago de Compostela, Spain, 2006.

[135] S. Silva, J. Madeira, and B. Sousa Santos. PolyMeCo — a polygonal mesh comparison tool. In *Proc. 9th International Conference in Information Visualization (IV05)*, pages 842–847, 2005.

[136] S. Silva, F. Silva, J. Madeira, and B. Sousa Santos. Evaluation of mesh simplification algorithms using PolyMeCo: A case study. In *Proc. SPIE Vol. 6495, 54950D, Visualization and Data Analysis 2007*, San José, California, USA, 2007.

[137] S. Silva, B. Sousa Santos, C. Ferreira, and J. Madeira. Comparison of methods for the simplification of mesh models using quality indices and an observer study. In *Proc. SPIE vol 6492, 64921L, Human Vision and Electronic Imaging XII*, San José, California, USA, 2007.

[138] S. Silva, B. Sousa Santos, J. Madeira, and C. Ferreira. Comparing three methods for simplifying mesh models of the lungs: An observer test to assess perceived quality. In *Proc. SPIE 2005 Vol. 5749, Image Perception, Observer Performance, and Technology Assessment*, pages 4847–495, 2005.

[139] H. Song, N. Cho, and J. Kim. Robust watermarking of 3D mesh models. In *Proc. IEEE Int. Workshop on Multimedia and Signal Processing*, pages 332–335, 2002.

[140] O. Sorkine. Laplacian mesh processing. In *EUROGRAPHICS 2005 — State-of-the-Art Report*, Dublin, Ireland, 2005.

[141] O. Sorkine, D. Cohen-Or, and S. Toledo. High-pass quantization for mesh encoding. In *Proc. Eurographics Symposium on Geometry Processing*, pages 42–51, 2003.

[142] O. Sourina, A. Sourin, and T. Howe. Orthopaedic surgery simulation. *World Scientific (in press)*, 2006.

[143] B. Sousa Santos, S. Silva, C. Ferreira, and J. Madeira. Comparison of methods for the simplification of mesh models of the lungs using quality indices and an observer study. In *Proc. 3rd International Conference on Medical Information Visualization – Biomedical Visualization (MediVis05)*, pages 15–21, 2005.

[144] C. Sun and J. Sherrah. 3D symmetry detection using the Extended Gaussian Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):164–168, 1997.

[145] T. Surazhsky, E. Magid, O. Soldea, G. Elber, and E. Rivlin. A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Proc. IEEE International Conference on Robotics and Automation (ICRA2003)*, pages 1021–1026, 2003.

[146] G. Sussner, G. Greiner, and S. Augustiniack. Interactive examination of surface quality on car bodies. *Computer-Aided Design*, 36(5):425–436, 2004.

[147] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedra approximation. In *Proc. 5th International Conference on Computer Vision*, pages 902–907, 1995.

[148] G. Taubin. A signal processing approach for fair surface design. In *Proc. SIGGRAPH 1995*, pages 351–358, 1995.

[149] G. Taubin. Geometric signal processing on polygonal meshes. In *EUROGRAPHICS 2000 — State-of-the-Art Report*, Interlaken, Switzerland, 2000.

[150] G. Taubin. Linear anisotropic mesh filtering. Technical report, IBM, T. J. Watson Research Center, 2001.

[151] W. Tedford, S. Berquist, and W. Flynn. The size-color illusion. *The Journal of General Psychology*, 97:145–150, 1977.

[152] H. Theisel, C. Rössl, R. Zayer, and H.-P. Seidel. Normal based estimation of the curvature tensor for triangular meshes. In *Proc. of the 12th Pacific Conference on Computer Graphics and Applications*, pages 288–297, 2004.

[153] B. Trumbo. Theory for coloring bivariate statistical maps. *The American Statistician*, 35(4):220–226, 1981.

[154] G. Turk. Re-tiling polygonal surfaces. In *Proc. SIGGRAPH 1992*, pages 55–64, 1992.

[155] S. Valette and P. Prost. Wavelet-based multiresolution analysis of irregular surface meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):113–122, 2004.

[156] O. van Kaick and H. Pedrini. A comparative evaluation of metrics for fast mesh simplification. *Computer Graphics Forum*, 25(2):197–210, 2006.

[157] C. Ware. Color sequences for univariate maps: Theory, experiments, and principles. *IEEE Computer Graphics & Applications*, 8(5):41–49, 1988.

[158] K. Watanabe and A. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, 20(3):385–392, 2001.

[159] B. Watson, A. Friedman, and A McGaffey. Measuring and predicting visual fidelity. In *Proc. SIGGRAPH 2001*, pages 213–220, 2001.

[160] K. Weiler. Edge-based data structures for solid modeling in curved surface environments. *IEEE Computer Graphics & Applications*, 5(1):21–40, 1985.

[161] J. Wu and L. Kobbelt. A stream algorithm for the decimation of massive meshes. In *Proc. Graphics Interface 2003*, pages 185–192, 2003.

[162] J. Wu and L. Kobbelt. Efficient spectral watermarking of large meshes with orthogonal basis functions. *The Visual Computer*, 21(8):848–857, 2005.

[163] H. Yee, S. Pattanaik, and D. Greenberg. Spatiotemporal sensivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20: 39–65, 2001.

[164] K. Yin, Z. Pan, J. Shi, and D. Zhang. Robust mesh watermarking based on multiresolution processing. *Computers & Graphics*, 25(3):409–420, 2001.

[165] S. Zafeiriou, A. Tefas, and P. Ioannis. Blind robust watermarking schemes for copyright protection of 3D mesh objects. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):596–607, 2005.

[166] L. Zhou and A. Pang. Metrics and visualization tools for surface mesh comparison. In *Proc. SPIE 2001 Vol. 4302, Visual Data Exploration and Analysis VIII*, pages 99–110, 2001.

[167] D. Zorin, P. Schröder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000.